

Cooperative Regenerating Codes

Kenneth W. Shum and Yuchong Hu

Abstract—One of the design objectives in distributed storage system is the minimization of the data traffic during the repair of failed storage nodes. By repairing multiple failures simultaneously and cooperatively, further reduction of repair traffic is made possible. We give a closed-form expression of the optimal tradeoff between the repair traffic and the amount of storage in each node for cooperative repair. We also show the existence of cooperative regenerating codes matching the points on the fundamental tradeoff curve. Specially, we show the existence of linear cooperative regenerating codes for functional repair, with an explicit bound on the required finite field size. Two families of explicit constructions are given.

Index Terms—Distributed Storage System, Regenerating Codes, Decentralized Erasure Codes, Network Coding, Submodular Flow.

I. INTRODUCTION

In order to provide high data reliability, distributed storage systems disperse data with redundancy to multiple storage nodes against node failures. There are two common redundancy schemes, *replication* employed by the current Google file system [1] and *erasure coding* used in Oceanstore [2] and TotalRecall [3]. Although replication-based scheme is easy to manage, it has much lower storage efficiency than erasure coding [4]. With (n, k) maximal-distance separable codes, a data file is encoded and distributed to n storage nodes, any k of which can reconstruct the original file. The data file remains intact even though some storage nodes may fail.

In case of node failures, we need to regenerate new nodes (called the *newcomers*) to replace the failed nodes. The newcomers are regenerated by downloading some data from the surviving nodes. The required traffic for repairing one single failed node, called *repair-bandwidth*, is another metric in measuring the system performance, which is essential in bandwidth-limited storage networks. In the pioneering work of Dimakis *et al.* [5], a class of erasure codes, called *regenerating codes*, is introduced to reduce the repair-bandwidth.

The repair based on regenerating codes can be roughly be divided into two classes. In the first class, called *exact repair*, the content of the newcomers are exactly the same as the content of the failed nodes. The second class is called *functional repair*. With functional repair, the content of the newcomers are not necessarily identical to the failed nodes, but the property that a data collector connecting to any subset of k nodes is able to decode the data file is preserved. On the one hand, in [5], the functional repair problem of minimizing the

repair-bandwidth is equivalent to a single-source multi-casting problem in network coding theory [6], [7], and a optimal tradeoff between the repair-bandwidth and the amount of data stored in each node is derived. Some studies on constructions of regenerating codes can be found in [8]–[19].

Most of the studies on regenerating codes in the literature focus on single-failure recovery for each repair. However, in large-scale distributed storage systems, a multiple-failure repair is the norm rather than the exception. For instance, in some practical systems such as TotalRecall [3], a recovery process is triggered only after the total number of failed nodes has reached a predefined threshold. In peer-to-peer storage systems with high churn rate, nodes may join and leave the system in batch. This can also be regarded as multiple node failures.

In this paper, we address the problem of repairing multiple node failures simultaneously, with the feature of data exchange among the newcomers enabled. This is called *cooperative repair* or *collaborative repair*, and is first introduced in [20]. We will call a regenerating code with cooperative repair a *cooperative regenerating code*. It is shown that repair-bandwidth can be further reduced if the node regeneration is performed jointly, instead of separately. In [21], a special class of cooperative regenerating codes is proposed, in which the newcomers can select survival nodes for repairing flexibly. In [20], [21], only the special case of minimum storage per node is considered. Studies of cooperative regenerating codes for minimum storage or minimum repair-bandwidth can be found in [22] and [23].

Using the information flow graph for cooperative repair, we derive the fundamental tradeoff between the storage per node and the repair-bandwidth. However, the size of the information flow graph is unbounded; there are infinitely many data collectors and arbitrary number of node failures. Existing algorithms for network code construction, such as the Jaggi-Sander *et al.*'s algorithm [24], assume that the graph is finite, and requires that the finite field size grows as the number of sink nodes increases. We therefore cannot apply Jaggi-Sander *et al.*'s algorithm directly to construct cooperative regenerating codes, unless we truncate the infinite information flow graph to a finite subgraph. This impose a maximum on the number of repairs and it is not obvious whether there exists cooperative regenerating code that can support arbitrary number of repairs without re-starting the system. Nevertheless, in the single-loss case, Wu in [25] succeeded in showing, by exploiting the structure of the information flow graph, that we can work over a fixed finite field and sustain the distributed storage system for indefinitely many repairs. In this paper, we generalize the results in [25] to cooperative repair.

K. W. Shum and Y. Hu are with Institute of Network Coding, the Chinese University of Hong Kong, Shatin, Hong Kong. Email: {wkshum,ychu}@inc.cuhk.edu.hk.

This work was partially supported by a grant from the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08).

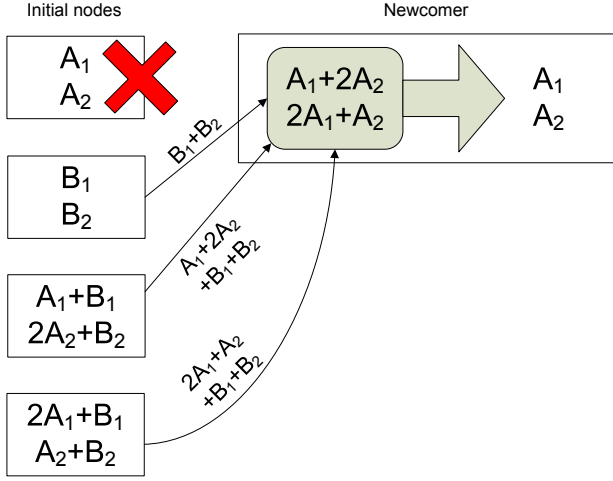


Fig. 1. Repairing a single node failure.

A. An Example of Cooperative Repair

We examine the following example taken from [26]. Four native data packets A_1 , A_2 , B_1 and B_2 are distributed to four storage nodes. Each of four storage nodes stores two packets. The first one stores A_1 and A_2 , the second stores B_1 and B_2 . The third node contains two packets $A_1 + B_1$ and $2A_2 + B_2$, and the last node contains $2A_1 + B_1$ and $A_2 + B_2$. Here, a packet is interpreted as an element in a finite field, and addition and multiplication are finite field operations. We can take $GF(5)$, the finite field of five elements, as the underlying finite field in this example. It can be readily checked that any data collector connecting to any two storage nodes can decode the four original packets.

Suppose that the first node fails. The naive method to repair the first node is to first reconstruct the four packets by connecting to any other two nodes, from which we can recover the two required packets A_1 and A_2 . Four packet transmissions are required in the naive method. The repair-bandwidth can be reduced from four packets to three packets by making three connections as in Fig. 1. Each of the three remaining nodes adds the stored packets and sends the sum to the newcomer, who can then subtract off $B_1 + B_2$ and obtain $A_1 + 2A_2$ and $2A_1 + A_2$. The packets A_1 and A_2 can now be solved readily. Hence, the newcomer can be regenerated exactly by sending three packets.

If two storage nodes fail simultaneously, four packets per newcomer are required if the newcomers are repaired separately (see Fig. 2). Each of the newcomers has to download four packets from the two surviving nodes, reconstruct packets A_1 , A_2 , B_1 and B_2 , and re-encode the desired packets. However, if exchange of data among the two newcomers is allowed, the total repair-bandwidth can be reduced from eight packets to six packets (see Fig. 3). The first newcomer gets A_1 and $A_1 + B_1$, while the second newcomer gets A_2 and $2A_2 + B_2$. The first newcomer then figures out B_1 and $2A_1 + B_1$ by taking the difference and the sum of the two inputs. The packet B_1 is stored and $2A_1 + B_1$ is sent to the second newcomer. Likewise, the second newcomer computes B_2 and $A_2 + B_2$, stores $A_2 + B_2$ and sends B_2 to the first

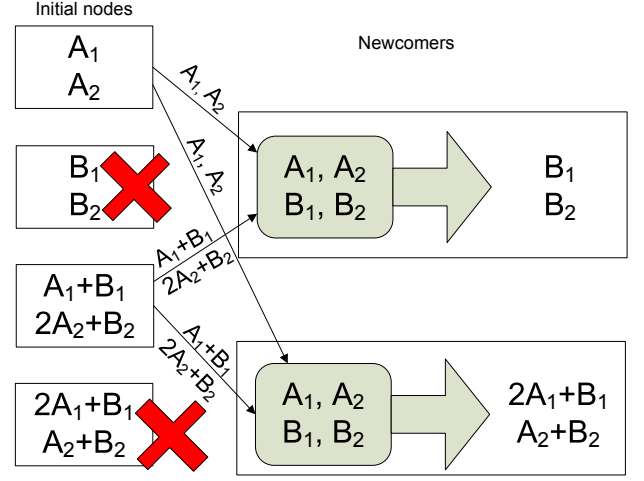


Fig. 2. Individual repair of multiple failures.

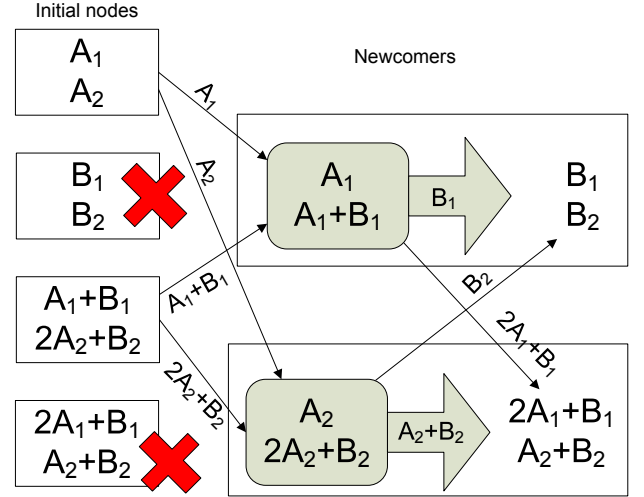


Fig. 3. Cooperative regeneration of multiple failures.

newcomer. The two newcomers are regenerated exactly with six packet transmissions. This example illustrates the potential benefit of cooperative repair.

B. Formal Definition of Cooperative Repair

Let \mathcal{Q} be an alphabet set of size q . We will call an element in \mathcal{Q} a *symbol*, and use a symbol as the unit of data. The data is regarded as a B -tuple $\mathbf{m} \in \mathcal{Q}^B$, with each component drawn from \mathcal{Q} . The distributed storage system consists of n nodes, and each node stores α symbols. We index the storage nodes by $\{1, 2, \dots, n\}$. There is an unlimited number of *data collectors* who want to download this data file from the storage nodes.

Time is divided into stages. We index the stages by non-negative integers. For $t \geq 0$, let the content of the i -th node in the t -th stage be denoted by an α -tuple $\mathbf{x}(t, i) \in \mathcal{Q}^\alpha$. The distributed storage system is initialized at stage 0 by setting $\mathbf{x}(0, i) = e_i(\mathbf{m})$ for $i = 1, 2, \dots, n$, where $e_i : \mathcal{Q}^B \rightarrow \mathcal{Q}^\alpha$ is an encoding function.

For a subset \mathcal{S} of $\{1, 2, \dots, n\}$, we let

$$\mathbf{x}(t, \mathcal{S}) := \{\mathbf{x}(t, i) : i \in \mathcal{S}\}$$

be the content of the storage nodes indexed by \mathcal{S} . The design objective is two-fold.

- 1) *File retrieval*. In each stage, a data collector can reconstruct the data file by connecting to any k out of the n storage nodes. This property is called the (n, k) *recovery property*. Mathematically, this means that for any k -subset \mathcal{S} of $\{1, 2, \dots, n\}$ and $t \geq 0$, there is a decoding function

$$f_{t,\mathcal{S}} : \mathcal{Q}^{k\alpha} \rightarrow \mathcal{Q}^B$$

such that $f_{t,\mathcal{S}}(\mathbf{x}(t, \mathcal{S})) = \mathbf{m}$ for all $\mathbf{m} \in \mathcal{Q}^B$.

- 2) *Multi-node recovery*. When the number of node failures in stage $s - 1$ reaches a threshold, say r , we jointly regenerate r newcomers, and advance to stage s . For $s = 1, 2, 3, \dots$, let \mathcal{R}_s be the set of r storage nodes which fail in stage $s - 1$ and are regenerated in stage s . The set \mathcal{R}_s contains r elements in $\{1, 2, \dots, n\}$. For each storage node $i \in \mathcal{R}_s$, let $\mathcal{H}_{s,i}$ be the d storage nodes in stage $s - 1$, called the *helpers*, from which data is downloaded to node i in stage s . The set $\mathcal{H}_{s,i}$ is a subset of $\{1, 2, \dots, n\} \setminus \mathcal{R}_s$ with cardinality d . Different newcomers may connect to different sets of d helpers. The repair procedure is divided into two phases.

In the first phase, each of the r newcomers download β_1 symbols from the d helpers. For $i \in \mathcal{R}_s$ and $j \in \mathcal{H}_{s,i}$, the symbols sent from node j to newcomer i is denoted by $g_{s,j,i}(\mathbf{x}(s - 1, j))$, where

$$g_{s,j,i} : \mathcal{Q}^\alpha \rightarrow \mathcal{Q}^{\beta_1}$$

is an encoding function.

In the second phase, the r newcomers exchange data among themselves. Every newcomer send β_2 symbols to each of the other $r - 1$ newcomers. For $i_1, i_2 \in \mathcal{R}_s$, let

$$g'_{s,i_1,i_2} : \mathcal{Q}^{d\beta_1} \rightarrow \mathcal{Q}^{\beta_2}$$

be the encoding function in the second phase, and

$$\mathbf{y}(s, i_1, i_2) = g'_{s,i_1,i_2}(\{g_{s,j,i_1}(\mathbf{x}(s - 1, j)) : j \in \mathcal{H}_{s,i_1}\})$$

be the symbols sent from newcomer i_1 to newcomer i_2 .

For each $i \in \mathcal{R}_s$, the content of the regenerated node i , $\mathbf{x}(s, i)$ is obtained by applying a mapping

$$h_{s,i} : \mathcal{Q}^{d\beta_1 + (r-1)\beta_2} \rightarrow \mathcal{Q}^\alpha$$

to $g_{s,j,i}(\mathbf{x}(s - 1, j))$ for $j \in \mathcal{H}_{s,i}$ and $\mathbf{y}(s, i_1, i)$ for $i_1 \in \mathcal{R}_s \setminus \{i\}$.

For those storage nodes that do not fail in stage $s - 1$, the content of them do not change, i.e., $\mathbf{x}(s, i) = \mathbf{x}(s - 1, i)$ for $i \notin \mathcal{R}_s$.

A *cooperative regenerating code*, or a *cooperative regeneration scheme*, is a collection of encoding and decoding functions e_i , $f_{t,\mathcal{S}}$, $g_{s,j,i}$, g'_{s,i_1,i_2} , and $h_{s,i}$, such that the (n, k) recovery property holds in all stages $t \geq 0$, and for all possible failure patterns \mathcal{R}_s and all choices of helper sets $\mathcal{H}_{s,i}$, $s \geq 1$.

A few more definitions and remarks are in order.

- If each storage node contains B/k symbols, then the regenerating code is said to have the *maximal-distance separable* (MDS) property.
- If $\mathbf{x}(t + 1, i) = \mathbf{x}(t, i)$ for all $t \geq 0$ and $i \in \{1, 2, \dots, n\}$, then the regenerating code is said to be *exact*.
- The *repair-bandwidth* per newcomer is denoted by

$$\gamma := d\beta_1 + (r - 1)\beta_2.$$

- The encoding functions $g_{s,j,i}$, g'_{s,i_1,i_2} , and $h_{s,i}$ depend on the indices of the failed nodes, \mathcal{R}_s , and the indices of the helper nodes, $\mathcal{H}_{s,i}$, or possibly depend on \mathcal{R}_t and $\mathcal{H}_{t,i}$ for $t \leq s$, i.e., the cooperative regeneration scheme is causal. For the ease of notation, this dependency is suppressed.
- The encoding and decoding is performed over a fixed alphabet set \mathcal{Q} in all stages.
- A pair $(\gamma/B, \alpha/B)$ is called an *operating point*. The first coordinate is ratio of the repair-bandwidth γ relative to the file size B , and the second coordinate is the ratio of the storage per node α relative to the file size. An operating point $(\tilde{\gamma}, \tilde{\alpha})$ is said to be *admissible* if there is a cooperative regeneration scheme over an alphabet set \mathcal{Q} with parameters B , α , β and γ , such that $(\tilde{\gamma}, \tilde{\alpha}) = (\gamma/B, \alpha/B)$. (The tildes indicate that the corresponding variables are normalized by the file size B .)
- For given d , k and r , let $\mathcal{C}(d, k, r)$ be the closure of all admissible operating points achieved by cooperative regenerating codes with parameters d , k and r . We call $\mathcal{C}(d, k, r)$ the *admissible region*. If the parameters d , k and r are clear from the context, we will simply write \mathcal{C} .
- For fixed storage α per node, we choose the two parameters β_1 and β_2 such that the repair-bandwidth γ is minimized. For given α and B , we let

$$\gamma^*(\tilde{\alpha}) := \min\{x : (x, \tilde{\alpha}) \in \mathcal{C}(d, k, r)\} \quad (1)$$

- In single-loss failure model ($r = 1$), it is shown in [5] that we only need to consider $d \geq k$ without loss of generality. In multiple-loss failure model ($r > 1$), there is no a priori reason why d cannot be strictly less than k . However, the mathematics for the case $d \geq k$ is simpler and more tractable. In this paper, we will assume that $d \geq k$. We will also assume that $k \geq 2$, because cooperative regenerating code with $k = 1$ is a trivial case.

We summarize the notations as follows:

- B : The size of the source file.
- n : The total number of storage nodes.
- d : Each newcomer connects to d surviving nodes.
- k : Each data collector connects to k storage nodes.
- r : The number of nodes repaired simultaneously.
- α : Storage per node.
- β_1 : Repair-bandwidth per newcomer in the 1st phase.
- β_2 : Repair-bandwidth per newcomer in the 2nd phase.
- γ : Total repair-bandwidth per newcomer.

C. Main Results and Organization

The main result of this paper is a closed-form expression for the region $\mathcal{C}(d, k, r)$. The statement of the main theorem requires the following notations.

Definitions: For $j = 2, 3, \dots, k$, define

$$\tilde{\alpha}_j := \frac{2(d - k + j) + r - 1}{D_j}, \quad (2)$$

$$\tilde{\gamma}_j := \frac{2d + r - 1}{D_j}, \quad (3)$$

where D_j is a short-hand notation for

$$D_j := k(2d - 2k + 2j + r - 1) - j(j - 1) \quad (4)$$

For $\ell = 0, 1, \dots, \lfloor k/r \rfloor$, define

$$\tilde{\alpha}'_\ell := \frac{d + r(\ell + 1) - k}{D'_\ell}, \quad (5)$$

$$\tilde{\gamma}'_\ell := \frac{d + r - 1}{D'_\ell}, \quad (6)$$

where

$$D'_\ell := k(d + r(\ell + 1) - k) - r^2\ell(\ell + 1)/2. \quad (7)$$

Definitions: For non-negative integer j and positive integer m , let

$$\Delta_{j,m} := \lfloor j/m \rfloor m^2 + (j - \lfloor j/m \rfloor m)^2. \quad (8)$$

For $j = 1, 2, \dots, k$, let

$$\mu(j) := \begin{cases} \frac{j(d-k) + (j^2 + \Delta_{j,m})/2}{jm - \Delta_{j,m}} & \text{if } \Delta_{j,m} < jm, \\ \infty & \text{if } \Delta_{j,m} = jm. \end{cases}$$

Let $\mu(0) := 0$.

We note that $\Delta_{0,m} = 0$, $\Delta_{1,m} = 1$ for all $m \geq 1$. Also, for $j \geq 2$ and $m \geq 1$,

$$j < \Delta_{j,m} \leq jm.$$

Equality $\Delta_{j,m} = jm$ holds if and only if j is divisible by m . (The quantities $\lfloor j/m \rfloor$ and $j - \lfloor j/m \rfloor m$ are respectively the quotient and the remainder when we divide j by m . The value of $\Delta_{j,m}$ can be interpreted as the maximum value of $\sum_{i=1}^j x_i^2$ subject to the constraints $\sum_{i=1}^j x_i = j$ and $0 \leq x_i \leq m$ for all i . The motivation for the definition of $\mu(j)$ will be given in Section III.)

Theorem 1. The admissible region $\mathcal{C}(d, k, r)$ is equal to the convex hull of the union of

$$\left\{ (\tilde{\gamma}_j, \tilde{\alpha}_j) : j = 2, 3, \dots, k-1, d \leq (r-1)\mu(j) \right\}, \quad (9)$$

$$\left\{ (\tilde{\gamma}'_{\lfloor j/r \rfloor}, \tilde{\alpha}'_{\lfloor j/r \rfloor}) : j = 2, 3, \dots, k-1, d > (r-1)\mu(j) \right\}, \quad (10)$$

$$\left\{ \left(\frac{d+r-1}{k(d+r-k)} + c, \frac{1}{k} \right) : c \geq 0 \right\}, \quad (11)$$

and

$$\left\{ \left(\frac{2d+r-1}{k(2d+r-k)}, \frac{2d+r-1}{k(2d+r-k)} + c \right) : c \geq 0 \right\}. \quad (12)$$

When $r = 1$, we define $0 \cdot \infty = \infty$ in (9) and (10).

We note that each of the sets in (9) and (10) contains at most $k-2$ points. The sets in (11) and (12) are horizontal and vertical rays respectively.

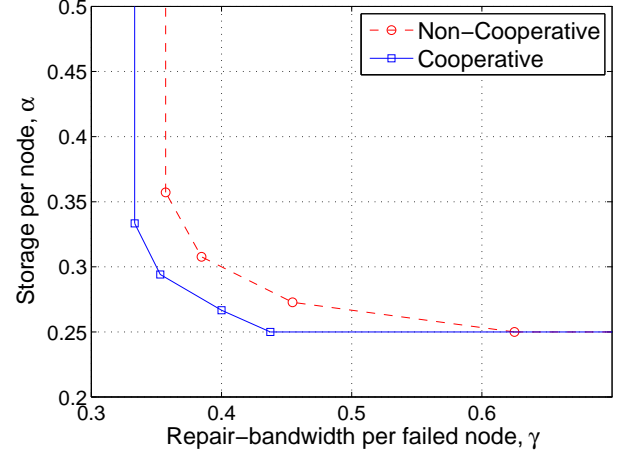


Fig. 4. Tradeoff between storage and repair bandwidth ($B = 1$, $d = 5$, $k = 4$, $r = 3$).

There are two particular operating points of special interest. The first one,

$$(\tilde{\gamma}_{\text{MSCR}}, \tilde{\alpha}_{\text{MSCR}}) := (\tilde{\gamma}'_0, \tilde{\alpha}'_0) = \left(\frac{d+r-1}{k(d+r-k)}, \frac{1}{k} \right),$$

is called the *minimum-storage cooperative regenerating* (MSCR) point. This point is the end point of the half-line (11).

The second one,

$$(\tilde{\gamma}_{\text{MBCR}}, \tilde{\alpha}_{\text{MBCR}}) := (\tilde{\gamma}_k, \tilde{\alpha}_k) = \frac{2d+r-1}{k(2d+r-k)}(1, 1),$$

is called the *minimum-bandwidth cooperative regenerating* (MBCR) point. This point is the end point of the half-line in (12).

An operating point $(\tilde{\gamma}_1, \tilde{\alpha}_1)$ is said to *Pareto-dominate* another point $(\tilde{\gamma}_2, \tilde{\alpha}_2)$ if $\tilde{\gamma}_1 \leq \tilde{\gamma}_2$ and $\tilde{\alpha}_1 \leq \tilde{\alpha}_2$. An operating point $(\tilde{\gamma}, \tilde{\alpha})$ is called *Pareto-optimal* if it is in $\mathcal{C}(d, k, r)$ and not Pareto-dominated by other operating points in $\mathcal{C}(d, k, r)$. In other words, a point $(\tilde{\gamma}, \tilde{\alpha})$ in the admissible region is Pareto-optimal if, for any other operating point $(\tilde{\gamma}', \tilde{\alpha}')$ in the admissible region, we have either $\tilde{\gamma}' > \tilde{\gamma}$ or $\tilde{\alpha}' > \tilde{\alpha}$, or both. The MSCR (resp. MBCR) point is the Pareto-optimal point with minimum $\tilde{\alpha}$ (resp. $\tilde{\gamma}$).

When $r = 1$, Theorem 1 reduces to the corresponding result for single-loss recovery in [5, Theorem 1]. Indeed, we have $\mu(j) = \infty$ for $j = 1, 2, \dots, k$ when $r = 1$. Using the convention that $0 \cdot \infty = \infty$, the set in (9) contains $k-2$ operating points

$$(\tilde{\gamma}_j, \tilde{\alpha}_j) = \frac{2}{2k(d-k+j) - j(j-1)}(d, d-k+j), \quad (13)$$

for $j = 2, 3, \dots, k-1$, while the set in (10) is empty. The extreme points of $\mathcal{C}(d, k, 1)$ are the points in (13) and

$$\left(\frac{d}{k(d+1-k)}, \frac{1}{k} \right), \left(\frac{2d}{k(2d+1-k)}, \frac{2d}{k(2d+1-k)} \right).$$

As a numerical example, we illustrate the admissible region $\mathcal{C}(5, 4, 3)$ (with parameters $d = 5$, $k = 4$, $r = 3$) in

Fig. 4. The solid line (marked by squares) is the boundary of the region $\mathcal{C}(5, 4, 3)$. The set in (9) contains two points, namely $(12/30, 8/30) = (0.4, 0.2667)$ and $(12/34, 10/34) = (0.3529, 0.2941)$, and the set in (10) is empty. The MSCR and MBCR points are respectively $(7/16, 1/4) = (0.4375, 0.25)$, and $(1/3, 1/3) = (0.3333, 0.3333)$. For comparison, we also plot in Fig. 4 the optimal tradeoff curve for single-failure repair (marked by circles).

In Section VII, two families of cooperative regenerating codes for exact repair are constructed explicitly. Both families have the property $d = k$. The first family matches the MSCR point, and has parameters $B = kr$, $n \geq d + r$, $\alpha = r$ and $\gamma = d + r - 1$. The second family matches the MBCR point and has parameters $B = k(k + r)$, $n = d + r$ and $\alpha = \gamma = 2d + r - 1$. We define the *storage efficiency* as the number of symbols in the data file divided by the total number of symbols in the n storage node. The first (resp. second) construction yields regenerating codes with storage efficiency k/n (resp. $k/(2k + r - 1)$).

In Section II, we define the information flow graph, and state some definitions and theorems from combinatorial optimization which will be used in Section V. In Section III, we give a lower bound on repair-bandwidth with cooperative recovery. In The lower bound is expressed in terms of a linear programming problem. In Section IV, we derive some properties of the linear programming problem, and discuss two kinds of feasible solutions of the linear programming problem. In Section V we show that the lower bound is tight. We prove in Section VI that we can construct functional-repair linear network codes over a fixed finite field, which match this lower bound on repair-bandwidth. The two explicit constructions for exact-repair cooperative regenerating codes mentioned in the previous paragraph are given in Section VII. Appendix A discuss the scenario where the download traffic may not be symmetric. Some of the longer proofs are relegated to the remaining appendices.

II. PRELIMINARIES

A. Information Flow Graph and the Max-Flow Bound

We review the modeling of the repair process using information flow graph as in [20]. The information flow graph is divided into stages; each stage involves the recovery of r failed nodes. Given parameters n, k, d and r , any directed graph $G = (\mathcal{V}, \mathcal{E})$ which can be constructed according to the following procedure is called an *information flow graph*.

- There is one single source vertex S in stage -1 , representing the original data file.
- The n storage nodes after initialization are represented by n vertices in stage 0, called Out_i , for $i = 1, 2, \dots, n$.
- For each $j \in \mathcal{R}_s$, we put three vertices in stage s : In_j , Mid_j and Out_j . For each $j \in \mathcal{R}_s$, there is a directed edge from In_j to Mid_j and a directed edge from Mid_j to Out_j . We draw d directed edges from the d nodes in $\mathcal{H}_{s,j}$ in stage $s - 1$ to node j in stage s . Namely, for each $i \in \mathcal{H}_{s,j}$, we put a directed edge from Out_i in stage $s - 1$ to In_j in stage s . The exchange of data among the

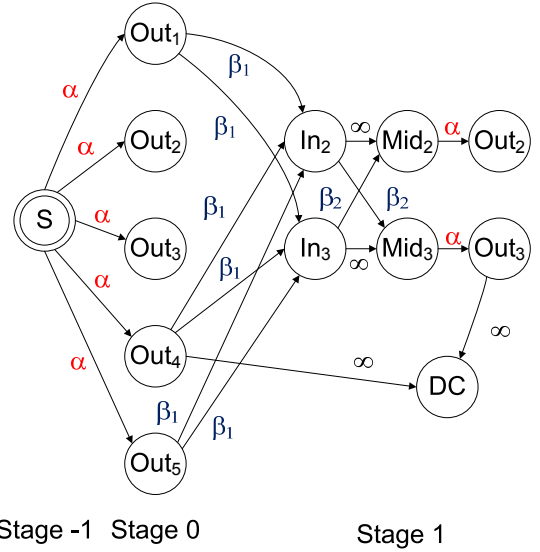


Fig. 5. An example of information flow graph $G(5, 3, 2, 2; \alpha, \beta_1, \beta_2)$. Nodes 2 and 3 are repaired in stage 1 ($\mathcal{R}_1 = \{2, 3\}$).

r newcomers are modeled by putting a directed from In_i to Mid_j for all pair of distinct i and j in \mathcal{R}_s .

- A data collector in stage s , called DC, is connected to k “out” nodes in the s -th or earlier stages.

We assign capacities to the edges as follows.

- The capacity of an edge terminating at an “out” node is α . This model the storage requirement in each storage node.
- The capacity of an edge from an “in” node to a “mid” node is infinity. The transfer of data is inside the newcomer and does not contribute to the repair-bandwidth
- The capacity from Out_i in stage $s - 1$ to In_j in stage s is β_1 , for $i \in \mathcal{H}_{s,j}$. This signifies the amount of data sent from Out_i to In_j in the first phase of the repair process. For the second phase, the edge from In_j to Mid_ℓ in stage s , for $j, \ell \in \mathcal{R}_s$ with $j \neq \ell$, is assigned a capacity of β_2 . (We use superscript (s) to indicate that the variable is associated with stage s .)
- The edges terminating at a data collector are all of infinite capacity.

The information flow graph so constructed is a directed acyclic graph. The number of stages may be unlimited and the information flow graph may be an infinite graph. We will denote an information flow graph by $G(n, d, k, r; \alpha, \beta_1, \beta_2)$. If the values of parameters are understood from the context, we will simply write G . An example of information flow graph is shown in Fig. 5.

Definitions: Let H be a directed graph with non-negative real numbers, called the capacities, assigned to the edges in H . An (S, T) -flow in H is a function f from the edge set \mathcal{E} of H to the non-negative real numbers, such that for every edge $e \in \mathcal{E}$, $f(e)$ does not exceed the capacity of e , and for every vertex v in $\mathcal{V} \setminus \{S, T\}$, the sum of $f(e)$ over all incoming edges e is equal to the sum of $f(e')$ over all outgoing edges e' . The *value* of an (S, T) -flow f is defined as the sum of $f(e)$ over all directed edges e terminating at vertex T . A flow f is

called *integral* if $f(e)$ is an integer for all e . An (S, T) -cut is a partition $(\mathcal{W}, \bar{\mathcal{W}})$ of the vertex set \mathcal{V} of H such that $S \in \mathcal{W}$ and $T \in \bar{\mathcal{W}}$. The capacity of an (S, T) -cut $(\mathcal{W}, \bar{\mathcal{W}})$ is defined as the sum of capacities of the edges from \mathcal{W} to $\bar{\mathcal{W}}$.

The max-flow-min-cut theorem states that the minimal cut capacity is equal to the largest possible flow value.

Definitions: For a given data collector DC in the information flow graph G , we let

$$\text{maxflow}(G, \text{DC})$$

be the maximal flow value from the source vertex S to DC.

Even though the graph G may be infinite, the computation of the flow from the source vertex to a particular data collector DC in stage t only involves the subgroup of G from stage -1 to stage t . For each DC the problem of determining the max-flow reduces to a max-flow problem in a finite graph.

An example of flow in an information flow graph for $n = 6$, $d = 4$, $k = 3$, $r = 2$, $\alpha = 7$, $\beta_1 = 2$, $\beta_2 = 1$ is shown in Fig. 6. The edges with positive flow are labeled (and drawn in red color). This is indeed a flow with maximal value, because there is a cut with capacity 19 (shown as the dashed line in Fig. 6).

According to the max-flow bound of network coding [27] [28, Theorem 18.3], if all data collectors are able to retrieve the original file, then the file size B is upper bounded by

$$B \leq \min_{\text{DC}} \text{maxflow}(G, \text{DC}). \quad (14)$$

This gives an upper bound on the supported file size for a given information flow graph G . Since we want to build cooperative regenerating schemes that can repair any pattern of node failures, we take the minimum

$$B \leq \min_G \min_{\text{DC}} \text{maxflow}(G, \text{DC}). \quad (15)$$

over all information flow graphs $G(n, d, k, r; \alpha, \beta_1, \beta_2)$.

Definitions: For given parameters n, d, k, r , we denote by

$$\mathcal{C}_{\text{MF}}(d, k, r) \quad (16)$$

the set of operating points $((d\beta_1 + (r-1)\beta_2)/B, \alpha/B)$ which satisfy the bound in (15).

Any operating point not in $\mathcal{C}_{\text{MF}}(d, k, r)$ violates the max-flow bound, and hence cannot be admissible,

$$\mathcal{C}_{\text{AD}}(d, k, r) \subseteq \mathcal{C}_{\text{MF}}(d, k, r). \quad (17)$$

We note that for fixed α, β_1 and β_2 , if B satisfy (15), then (15), is satisfied for all B' between 0 and B . Hence, if $(\tilde{\gamma}, \tilde{\alpha}) \in \mathcal{C}_{\text{MF}}(d, k, r)$, then $(c\tilde{\gamma}, c\tilde{\alpha}) \in \mathcal{C}_{\text{MF}}(d, k, r)$ for all $c \geq 1$.

Definitions: Let

$$\gamma_{\text{MF}}^*(\tilde{\alpha}) := \min\{x : (x, \tilde{\alpha}) \in \mathcal{C}_{\text{MF}}(d, k, r)\}. \quad (18)$$

B. Combinatorial Optimization

Definitions: Let \mathbb{R} be the set of real numbers, and \mathcal{V} be a finite set. The cardinality of \mathcal{V} is denoted by $|\mathcal{V}|$. Let $2^{\mathcal{V}}$ be the set of all subsets of \mathcal{V} . A function $f : 2^{\mathcal{V}} \rightarrow \mathbb{R}$ is called *submodular* if it satisfies

$$f(S) + f(T) \geq f(S \cap T) + f(S \cup T) \quad (19)$$

for all $S, T \subseteq \mathcal{V}$. To show that a function f is submodular, it is sufficient to check

$$f(S \cup \{u\}) + f(S \cup \{v\}) \geq f(S) + f(S \cup \{u, v\})$$

for all subsets $S \subseteq \mathcal{V}$ and $u, v \in \mathcal{V}$ (See for example [29, Thm 44.1]). If (19) holds with equality for all subsets S and T , then f is called *modular*. A submodular function which is invariant under all permutations of \mathcal{V} is said to be *symmetric*. In other words, a submodular function f is symmetric if for all permutations π of \mathcal{V} , we have $f(\pi(S)) = f(S)$, where $\pi(S)$ denotes the image of S under the mapping π .

Several examples of modular functions are given as follows. For a given function $g : \mathcal{V} \rightarrow \mathbb{R}$, we use $g(S)$ as a short-hand notation for $\sum_{x \in S} g(x)$. It is easy to verify that the function $S \mapsto g(S)$ is a modular function. Let $a_1 \geq a_2 \geq \dots \geq a_{|\mathcal{V}|}$ be $|\mathcal{V}|$ real numbers sorted in non-increasing order such that $a_{i+1} - a_i \geq a_{i+2} - a_{i+1}$ holds for $i = 2, 3, \dots, |\mathcal{V}|$. Then, the function given by

$$f(S) := \begin{cases} 0 & \text{if } S = \emptyset \\ a_i & \text{if } |S| = i, \end{cases}$$

is a symmetric submodular function.

For a given vector $\mathbf{v} = [v_1 \ v_2 \ \dots \ v_n]$ with non-negative components, we sort the components of \mathbf{v} in non-increasing order and let the j -th largest component in \mathbf{v} be denoted by $v_{[j]}$, i.e., $v_{[1]} \geq v_{[2]} \geq \dots \geq v_{[n]}$. Let \mathbb{R}_+^n be the set of n -dimensional vector with non-negative components. Given two vectors $\mathbf{v} = [v_1 \ v_2 \ \dots \ v_n]$ and $\mathbf{u} = [u_1 \ u_2 \ \dots \ u_n]$ in \mathbb{R}_+^n , we say that \mathbf{v} is *majorized* by \mathbf{u} if

$$v_{[1]} + v_{[2]} + \dots + v_{[i]} \leq u_{[1]} + u_{[2]} + \dots + u_{[i]},$$

for $i = 1, 2, \dots, n-1$ and

$$v_{[1]} + v_{[2]} + \dots + v_{[n]} = u_{[1]} + u_{[2]} + \dots + u_{[n]}.$$

A submodular function f mapping subsets of $\{1, 2, \dots, n\}$ to \mathbb{R}_+ , with the additional properties that (i) $f(\emptyset) = 0$ and (ii) $f(S) \geq f(T)$ whenever $S \supseteq T$ is called a *rank function*. For a vector $\mathbf{x} = (x_1, \dots, x_n)$ in \mathbb{R}_+^n and a subset $S \subseteq \{1, 2, \dots, n\}$, we use the notation

$$\mathbf{x}(S) := \sum_{i \in S} x_i.$$

Given a rank function f , the set

$$\{\mathbf{x} \in \mathbb{R}_+^n : \mathbf{x}(S) \leq f(S), \forall S \subseteq \{1, 2, \dots, n\}\}$$

is called a *polymatroid*. The face of the polymatroid consisting of the points satisfying $\sum_{i=1}^n x_i = f(\{1, 2, \dots, n\})$ is called the *base-polymatroid* associated with the rank function f .

Lemma 2. Let $\mathbf{u} \in \mathbb{R}_+^{|\mathcal{V}|}$ be a vector with non-negative components. Then the function $f : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$ defined by

$$f(S) := \sum_{i=1}^{|S|} u_{[i]}$$

is a symmetric submodular function. The vectors in $\mathbb{R}_+^{|\mathcal{V}|}$ which are majorized by \mathbf{u} form a base-polymatroid with rank function f .

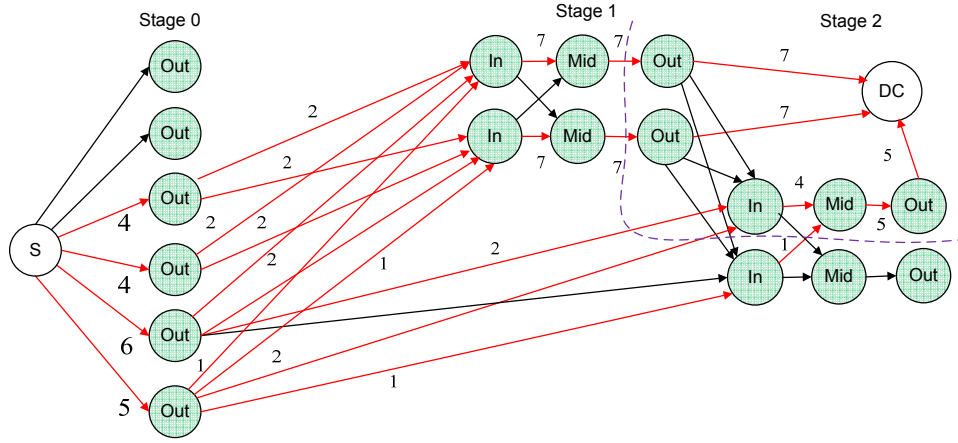


Fig. 6. An example of flow in an information flow graph. The parameters are $n = 6$, $d = 4$, $k = 3$, $r = 2$, $\alpha = 7$, $\beta_1 = 2$ and $\beta_2 = 1$. The labels of the edges indicate a flow on the graph (the arrows in red are assigned positive flow value and the arrows in black are assigned zero value). A cut with capacity 19 is illustrated by a dashed line.

The proof is straightforward and is omitted. We give one numerical example for Lemma 2. Let \mathbf{u} be the vector $[2 \ 2 \ 1]$. The vectors in \mathbb{R}_+^3 which are majorized by \mathbf{u} form a base-polymatroid

$$\{\mathbf{x} \in \mathbb{R}_+^3 : \mathbf{x}(S) \leq f(S), \forall S \subsetneq \{1, 2, 3\}, \\ \text{and } x_1 + x_2 + x_3 = 5\},$$

where

$$f(S) = \begin{cases} 0 & \text{if } S = \emptyset \\ 2 & \text{if } |S| = 1 \\ 4 & \text{if } |S| = 2 \\ 5 & \text{if } |S| = 3, \end{cases}$$

is a symmetric submodular function.

We need a generalization of the max-flow-min-cut theorem. The generalization is in terms of the notion of submodular flow, which we introduce below.

Definitions: Let $H = (\mathcal{V}, \mathcal{E})$ be a directed graph. For a given subset \mathcal{T} of \mathcal{V} , define the set of incoming edges and the set of out-going edges respectively by

$$\Delta_{\mathcal{T}}^{in} := \{e = (u, v) \in \mathcal{E} : u \notin \mathcal{T}, v \in \mathcal{T}\}, \\ \Delta_{\mathcal{T}}^{out} := \{e = (u, v) \in \mathcal{E} : u \in \mathcal{T}, v \notin \mathcal{T}\}.$$

Let $\phi : \mathcal{E} \rightarrow \mathbb{R}$ be a function on the set of edges of H . Define the *boundary* of a subset of vertices $\mathcal{T} \in \mathcal{V}$ by

$$\partial\phi(\mathcal{T}) := \phi(\Delta_{\mathcal{T}}^{out}) - \phi(\Delta_{\mathcal{T}}^{in}).$$

The boundary of \mathcal{T} can be considered as the net flow leaving \mathcal{T} . Given a submodular function $f : 2^{\mathcal{V}} \rightarrow \mathbb{R}$, we say that a function $\phi : \mathcal{E} \rightarrow \mathbb{R}$ is an f -submodular flow, or simply *submodular flow*, if

$$\partial\phi(\mathcal{T}) \leq f(\mathcal{T}) \quad (20)$$

for all $\mathcal{T} \subseteq \mathcal{V}$. Let $\text{lb} : \mathcal{E} \rightarrow \mathbb{R} \cup \{-\infty\}$ and $\text{ub} : \mathcal{E} \rightarrow \mathbb{R} \cup \{\infty\}$ be respectively lower and upper bound on edges, with $\text{lb}(e) \leq \text{ub}(e)$ for each $e \in \mathcal{E}$. A submodular flow ϕ is said to be *feasible* if $\text{lb}(e) \leq \phi(e) \leq \text{ub}(e)$ for all $e \in \mathcal{E}$.

The following theorem, due to A. Frank [30] [31, Theorem 12.1.4], characterizes the existence of submodular flow. It is central in the proof of the main theorems in this paper.

Theorem 3 ([30]). *Suppose that ρ is a submodular function on the vertex set of a directed graph $(\mathcal{V}, \mathcal{E})$, satisfying $\rho(\emptyset) = \rho(\mathcal{V}) = 0$. There exists a feasible submodular flow if and only if*

$$\text{lb}(\Delta_{\mathcal{S}}^{out}) - \text{ub}(\Delta_{\mathcal{S}}^{in}) \leq \rho(\mathcal{S}) \quad (21)$$

for all subsets $\mathcal{S} \subseteq \mathcal{V}$. Moreover, if lb , ub and ρ are integer-valued, then there is a submodular flow which is integer-valued.

To see that the max-flow-min-cut theorem is a special case of Frank's theorem, we consider a weighted directed graph $H = (\mathcal{V}, \mathcal{E})$ with two distinguished vertices S and T . Define a function $g : \mathcal{V} \rightarrow \mathbb{R}$ by

$$g(x) = \begin{cases} B & \text{if } x = S, \\ -B & \text{if } x = T, \\ 0 & \text{otherwise,} \end{cases}$$

where B is a positive real number. For a subset \mathcal{W} of the vertex set, define

$$\rho(\mathcal{W}) = \sum_{x \in \mathcal{W}} g(x).$$

For $e \in \mathcal{E}$, let $\text{lb}(e) = 0$ and $\text{ub}(e)$ be the corresponding edge capacity. The function ρ so defined is modular and satisfies $\rho(\emptyset) = \rho(\mathcal{V}) = 0$. With this choice of submodular function ρ , a feasible ρ -submodular flow is equivalent to a flow with value B . Indeed, it is easy to see that an (S, T) -flow with value B is a feasible ρ -submodular flow. In the reverse direction, let ϕ be a feasible ρ -submodular flow. We have

$$B \leq -\partial\phi(\{T\}) = \partial\phi(\mathcal{V} \setminus \{T\}) = \sum_{v \neq T} \partial\phi(\{v\}) \\ \leq \partial\phi(\{S\}) \leq B.$$

The inequalities follow from the definition of the submodular function ρ ,

$$\partial\phi(\{x\}) \leq \begin{cases} B & \text{if } x = S \\ -B & \text{if } x = T \\ 0 & \text{if } S \neq x \neq T. \end{cases}$$

Since equalities hold in the above chain of inequalities, we have $\partial\phi(\{v\}) = 0$ for all vertices v other than S and T , i.e., ϕ satisfies the flow conservation property. The ρ -submodular flow ϕ is indeed a flow. From $-\partial\phi(\{T\}) = B$ we conclude that the flow has value B .

For a subset \mathcal{W} of the vertex set not containing S nor T , the condition in (21) is equivalent to requiring that the capacity of the cut $(\bar{\mathcal{W}}, \mathcal{W})$ is at least B . Hence, if all (S, T) -cuts have capacity at least B , then, by Frank's theorem, we can obtain a feasible ρ -submodular flow, which is an (S, T) -flow with value B . If there is a cut with capacity strictly less than B , then the condition (21) is violated by some subset \mathcal{S} of the vertex set, and thus there does not exist a feasible ρ -submodular flow.

III. A CUT-SET LOWER BOUND ON REPAIR-BANDWIDTH

Consider a data collector DC which downloads data from k storage nodes. By re-labeling the storage nodes, we can assume without loss of generality that the DC downloads from nodes 1 to k . Suppose that among these k nodes, ℓ_0 of them do not undergo any repair, and the remaining $k - \ell_0$ nodes are repaired in stage 1 to s for some positive integer s . For $j = 1, 2, \dots, s$, suppose that there are ℓ_j nodes which are repaired in stage j and connected to the data collector DC. We can check that $\ell_0 + \ell_1 + \dots + \ell_s = k$ and $1 \leq \ell_j \leq r$ (for $j \geq 1$). After some re-labeling, we assume that the ℓ_0 unrepaired nodes are node 1 to node ℓ_0 , the nodes which are repaired in stage 1 are node $\ell_0 + 1$ to node $\ell_0 + \ell_1$, and so on.

In the information flow graph, the data collector DC is connected to ℓ_j "out" vertices in stage j . A cut $(\mathcal{W}, \bar{\mathcal{W}})$ with $\bar{\mathcal{W}}$ consisting of the data collector DC, the ℓ_0 "out" vertices in stage 0 associated with nodes 1 to ℓ_0 , and

$$\bigcup_{i=\ell_{j-1}+1}^{\ell_j} \{\text{In}_i, \text{Mid}_i, \text{Out}_i\}$$

in stage j , for $j = 1, 2, \dots, s$, is called a cut of type

$$(\ell_0, \ell_1, \ell_2, \dots, \ell_s).$$

An example of cut $(\mathcal{W}, \bar{\mathcal{W}})$ of type $(2, 1, 1, 2)$ is shown in Fig. 7. Nodes 3 and 4 are repaired in stage 1, nodes 4 and 7 are repaired in stage 2, and nodes 5 and 6 are repaired in stage 3. The data collector connects to nodes 1 to 6. The vertices in $\bar{\mathcal{W}}$ are drawn in shaded color in Fig. 7.

Theorem 4. For any $(s+1)$ -tuples of integers $(\ell_0, \ell_1, \dots, \ell_s)$ satisfying $\sum_{j=0}^s \ell_j = k$ and $1 \leq \ell_j \leq r$ for all j , the file size B is less than or equal to

$$\ell_0 \alpha + \sum_{j=1}^s \left[\ell_j \left(d - \sum_{i=0}^{j-1} \ell_i \right) \beta_1 + \ell_j (r - \ell_j) \beta_2 \right]. \quad (22)$$

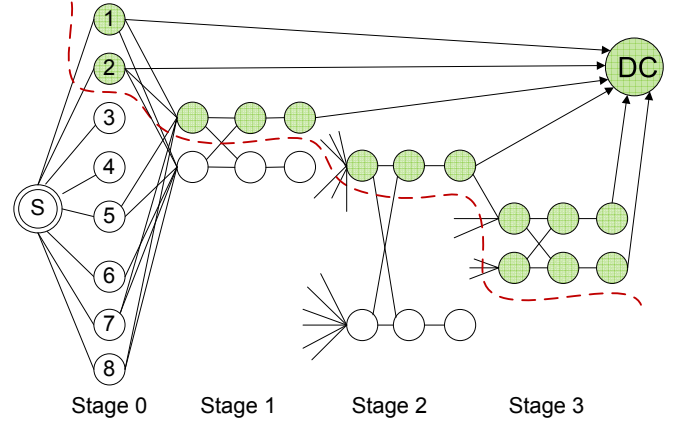


Fig. 7. A cut of type $(2, 1, 1, 2)$ in a distributed storage system with parameters $d = 6$, $k = 6$ and $r = 2$.

(The term $d - \sum_{i=0}^{j-1} \ell_i$ in (22) is nonnegative, because the summation of ℓ_i 's is no larger than k , and k is assumed to be less than or equal to d .)

Proof: It suffices to show that the capacity of a cut of type $(\ell_0, \ell_1, \dots, \ell_s)$ is larger than or equal to (22).

The sums of capacities of the edges terminating at the ℓ_0 "out" vertices in stage 0 is $\ell_0 \alpha$. This is the first term in (22). For $j = 1, 2, \dots, s$, consider an "in" vertex in stage j in $\bar{\mathcal{W}}$. Out of the d incoming edges to this "in" vertex, there may be as small as $d - \sum_{i=0}^{j-1} \ell_i$ edges which start from some "out" vertices in \mathcal{W} . The sum of the capacities of the edges from \mathcal{W} to the "in" vertices in stage j is larger than or equal to

$$\ell_j \left(d - \sum_{i=0}^{j-1} \ell_i \right) \beta_1.$$

This is the first term inside the square bracket. The second term in the square bracket is the sum of edge capacities to the "mid" vertices in $\bar{\mathcal{W}}$. ■

Theorem 5. If a data file of size B is supported by a cooperative regenerating code with parameters $n, d, k, r, \alpha, \beta_1$ and β_2 , then for $j = 0, 1, \dots, k$, we have

$$1 \leq (k - j) \frac{\alpha}{B} + j \left[d - k + \frac{j+1}{2} \right] \frac{\beta_1}{B} + j(r - 1) \frac{\beta_2}{B} \quad (23)$$

and

$$1 \leq (k - j) \frac{\alpha}{B} + \left[j(d - k) + \frac{j^2 + \Delta_{j,r}}{2} \right] \frac{\beta_1}{B} + (jr - \Delta_{j,r}) \frac{\beta_2}{B}, \quad (24)$$

where $\Delta_{j,r}$ is given in (8).

Proof: The upper bound in (23) comes from a cut of type

$$(k - j, \underbrace{1, 1, \dots, 1}_j).$$

There are $j+1$ components in the above vector. The derivation

of (23) follows from

$$\begin{aligned} & \sum_{j=1}^s \ell_j \left(d - \sum_{i=0}^{j-1} \ell_i \right) \\ &= (d - k + s) + (d - k + s - 1) + \cdots + (d - k + 1) \\ &= s \left[d - k + \frac{s+1}{2} \right]. \end{aligned}$$

The upper bound in (24) comes from a cut of type

$$(k - j, \underbrace{r, r, \dots, r}_Q, R),$$

where Q and R are defined respectively as the quotient and remainder when we divide j by r . (Q and R are integers satisfying $j = Qr + R$ and $0 \leq R < r$.) There $Q + 2$ components in the above vector.

Straightforward calculations show that the right-hand side of (24) is

$$\begin{aligned} & (k - j)\alpha + \left[d - k + j - \frac{(Q - 1)r}{2} \right] Qr\beta_1 \\ & + (d - k + j - Qr)R\beta_1 + R(r - R)\beta_2. \end{aligned}$$

Note that $\Delta_{j,r}$ is equal to $Qr^2 + R^2$. In terms of $\Delta_{j,r}$, the coefficient of β_2 is

$$R(r - R) = Rr - R^2 = (j - Qr)r - R^2 = jr - \Delta_{j,r},$$

and the coefficient of β_1 is

$$\begin{aligned} & \left[d - k + j - \frac{(Q - 1)r}{2} \right] Qr + (d - k + j - Qr)R \\ &= j(d - k + j) - \frac{Q^2r^2 - Qr^2 + QrR}{2} \\ &= j(d - k + j) - \frac{j^2 - Qr^2 - R^2}{2} \\ &= j(d - k) + \frac{j^2 + \Delta_{j,r}}{2}. \end{aligned}$$

This proves the inequality in (24). \blacksquare

Remark: (i) When $j = 0$, the two inequalities in (23) and (24) are identical and can be simplified to

$$B \leq k\alpha. \quad (25)$$

(ii) When $j = 1$, (23) and (24) are also identical and can be written as

$$B \leq (k - 1)\alpha + (d - k + 1)\beta_1 + (r - 1)\beta_2. \quad (26)$$

(iii) The coefficients of α , β_1 and β_2 in (23) and (24) are non-negative.

(iv) In the special case of single-loss repair, i.e., when $r = 1$, the coefficients of β_2 in (23) and (24) vanish.

Example: We can now show that the example of cooperative regenerating code mentioned in the introductory section is optimal. The system parameters are $B = 4$, $\alpha = 2$ and $d = k = r = 2$. After putting $j = 1$ and $j = 2$ in Theorem 5, the inequality in (24) become

$$\begin{aligned} 4 &\leq 2 + \beta_1 + \beta_2, \\ 4 &\leq 4\beta_1. \end{aligned}$$

This implies that $\beta_1 \geq 1$ and $\beta_2 \geq 1$. If we want to minimize the repair-bandwidth $\gamma = 2\beta_1 + \beta_2$, the optimal solution is attained at $(\beta_1, \beta_2) = (1, 1)$. The optimal repair-bandwidth is thus equal to 3. The above analysis also shows that if the repair-bandwidth is equal to 3, the values of β_1 and β_2 must both equal to 1. This is indeed the case in the example given in the introduction.

We note that the bounds in (23) and (24) depend on α , β_1 and β_2 through the ratio α/B , β_1/B and β_2/B . We can thus consider the normalized β_1/B and β_2/B as variables. We have the following linear programming problem.

Definitions: Let $\tilde{\alpha} := \alpha/B$, $\tilde{\beta}_1 := \beta_1/B$, $\tilde{\beta}_2 := \beta_2/B$, and $\tilde{\gamma} := \gamma/B$ be the normalized values of α , β_1 , β_2 and γ respectively. Consider the following optimization problem:

$$\text{Minimize } (d\tilde{\beta}_1 + (r - 1)\tilde{\beta}_2) \quad (27)$$

subject to the linear constraints in (23) and (24), for $j = 1, 2, \dots, k$, and $\tilde{\beta}_1, \tilde{\beta}_2 \geq 0$. This is a parametric linear programming problem with $\tilde{\alpha}$ as the parameter. Let

$$\gamma_{\text{LP}}^*(\tilde{\alpha}) \quad (28)$$

be the optimal value of this linear program, and

$$\mathcal{C}_{\text{LP}}(n, k, d, r) = \mathcal{C}_{\text{LP}} := \{(\tilde{\gamma}, \tilde{\alpha}) \in \mathbb{R}^2 : \tilde{\gamma} \geq \gamma_{\text{LP}}^*(\tilde{\alpha})\}. \quad (29)$$

At this point, we have established the following relationship

$$\mathcal{C}_{\text{AD}} \subseteq \mathcal{C}_{\text{MF}} \subseteq \mathcal{C}_{\text{LP}}. \quad (30)$$

The first inclusion follows from the max-flow bound in network coding. Since the number of sink nodes in the information flow graph is infinite, we cannot apply existing results in the construction of network codes, such as the Jaggi-Sanders *et al.*'s algorithm, to guarantee that we can work over a fixed alphabet set for infinitely many data collectors. The second inclusion follows from a weaker form of max-flow-min-cut bound in graph theory, namely, the value of any flow is no larger than the capacity of any cut, since we only consider some specific cuts in the information flow graph, not all possible cuts. In later sections, we will show that equalities hold in (30).

Example: Consider a cooperative-repair-based distributed storage system with parameters $d = 5$, $k = 4$ and $r = 3$. We have following constraints based on (23) and (24):

$$\begin{bmatrix} B \\ B \\ B \\ B \\ B \\ B \\ B \\ B \end{bmatrix} \leq \begin{bmatrix} 4 & 0 & 0 \\ 3 & 2 & 2 \\ 2 & 6 & 2 \\ 2 & 5 & 4 \\ 1 & 12 & 0 \\ 1 & 9 & 6 \\ 0 & 17 & 2 \\ 0 & 14 & 8 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \end{bmatrix}, \quad (31)$$

with the inequality compared componentwise. We normalize B to 1 unit and consider the case when $\alpha = 1/4$, i.e., the minimum-storage case. We minimize $d\beta_1 + (r - 1)\beta_2$ subject to $\beta_1, \beta_2 \geq 0$ and the constraints in (31) by linear programming. The linear constraints and the objective function are illustrated graphically in Fig. 8. The seven solid lines (in blue color) in Fig. 8 are the boundary of the half planes associated with the

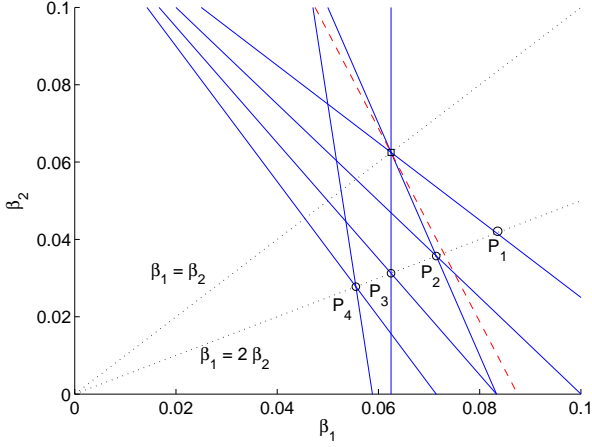


Fig. 8. Repair-bandwidth minimization as a linear program ($d = 5$, $k = 4$, $r = 3$, and $\alpha = 1/4$). The objective function $5\beta_1 + 2\beta_2$ is minimized at $\beta_1 = \beta_2 = 0.625$.

seven constraints (row 2 to row 8) in (31). The point P_1 is the intersection point of the straight line associated to row 2, i.e., $B = 3\alpha + 2\beta_1 + 2\beta_2$, and the line $\beta_1 = 2\beta_2$. The point P_2 is the intersection point of the straight lines associated to rows 3 and 4 in (31). The point P_3 is the intersection point of the straight lines associated to rows 5 and 6, and so on. The feasible region is the area to the right and above these seven straight lines. The optimal solution $\beta_1 = \beta_2 = 0.625$ is indicated by a square in Fig. 8, and the objective function is shown as a dashed line (in red) passing through this point. The repair-bandwidth cannot be less than

$$\gamma_{LP}^*(1/4) = (d + r - 1) \cdot 0.625 = (5 + 3 - 1) \cdot 0.625 = 4.375.$$

The bound in Theorem 5 is based on the assumption that the download traffic is homogeneous. In Appendix A, we show at the minimum-storage point, the relaxation of the homogeneity in download traffic does not help in further reducing the repair-bandwidth. In the remaining of this paper, we will assume that the download traffic is homogeneous.

IV. THE TWO TYPES OF OPERATING POINTS

In this section we explain the two sets of operating points in (9) and (10). For $j = 2, 3, \dots, k$, the point $(\tilde{\gamma}_j, \tilde{\alpha}_j)$ is said to be an *operating point of the first type*. For $\ell = 0, 1, \dots, \lfloor k/r \rfloor$, the point $(\tilde{\gamma}'_\ell, \tilde{\alpha}'_\ell)$ is said to be an *operating point of the second type*.

Definitions: For $j = 1, 2, \dots, k$, we let $L_j(\alpha)$ be the straight line in the β_1 - β_2 plane with equation

$$B = (k - j)\alpha + \left(j(d - k) + \frac{j^2 + \Delta_{j,r}}{2}\right)\beta_1 + (jr - \Delta_{j,r})\beta_2 \quad (24')$$

and $L'_j(\alpha)$ be the straight line with equation

$$B = (k - j)\alpha + \left(j(d - k) + \frac{j^2 + j}{2}\right)\beta_1 + (jr - j)\beta_2. \quad (23')$$

Here, α is treated as a fixed parameter.

We record some easy facts in the following lemma.

Lemma 6.

- 1) $\gamma_{LP}^*(\alpha) = \infty$ for $\alpha < B/k$, and is monotonically decreasing as α increases.
- 2) For $j = 1, 2, \dots, k$, the slope of the straight line $L'_j(\alpha)$ is

$$-\frac{d - k + (j + 1)/2}{r - 1},$$

and the magnitude is strictly less than $d/(r - 1)$ when $r \geq 2$.

- 3) If j is an integral multiple of r , then the slope of the straight line $L_j(\alpha)$ is infinite.
- 4) The line $L_1(\alpha)$ is identical to line $L'_1(\alpha)$. When $r = 1$, the line $L_j(\alpha)$ is identical to line $L'_j(\alpha)$, for $2 \leq j \leq k$.
- 5) For $r \geq 2$ and $2 \leq j \leq k$, the magnitude of the slope of $L_j(\alpha)$ is strictly larger than the magnitude of the slope of $L'_j(\alpha)$. $L_j(\alpha)$ and $L'_j(\alpha)$ intersect at a point lying on the line $\beta_1 = 2\beta_2$ in the β_1 - β_2 plane.

Proof:

(1) If $\alpha < B/k$, the constraint in (25) is violated. Thus $\gamma_{LP}^*(\alpha) = \infty$ for $\alpha < B/k$. As α increases, the feasible region of the linear program in Theorem 5 is enlarged, and thus the minimal value decreases.

(2) The line $L'_j(\alpha)$, whose equation is given in (23'), has magnitude

$$\frac{j(d - k) + (j^2 + j)/2}{jr - j} = \frac{d - k + (j + 1)/2}{r - 1}.$$

This fraction is strictly less than $d/(r - 1)$ for $r \geq 2$ and $j \leq k$.

(3) It follows from the fact that $\Delta_{j,r} = jr$ when j is a multiple of r .

(4) When $j = 1$, we have $\Delta_{j,r} = j = 1$ for all r . When $r = 1$, we have $\Delta_{j,r} = j$ for $j = 2, 3, \dots, k$.

(5) For $j = 2, 3, \dots, k - 1$, the determinant

$$\begin{vmatrix} j(d - k) + \frac{j^2 + \Delta_{j,r}}{2} & jr - \Delta_{j,r} \\ j(d - k) + \frac{j^2 + j}{2} & jr - j \end{vmatrix}$$

is equal to

$$j(\Delta_{j,r} - j)[d - k + (r + j)/2].$$

Since $\Delta_{j,r} > j$ for $j \geq 2$, and $d \geq k$ by assumption, the determinant is positive, and thus the magnitude of the slope of $L'_j(\alpha)$ is strictly larger than the magnitude of the slope of $L(\alpha)$. By subtracting (23') from (24'), we obtain $\beta_1 = 2\beta_2$ after some simplifications. ■

Definitions: For $j = 1, 2, 3, \dots, k$, let $P_j(\alpha)$ be the intersection point of $L_j(\alpha)$, $L'_j(\alpha)$, and on the line $\beta_1 = 2\beta_2$.

Lemma 7. For $j \geq 1$, the coordinates of $P_j(\alpha)$ is

$$\frac{B - (k - j)\alpha}{j(2d - 2k + r + j)}(2, 1). \quad (32)$$

Proof: Put $\beta_1 = 2\beta_2$ in (23'). ■

For $d = 5$, $k = 4$ and $r = 3$, the points $P_i(1/4)$, for $i = 1, 2, 3, 4$, are shown in Fig. 8. By part 5 of Lemma 6, we

can explicitly calculate their coordinates:

$$\begin{aligned} P_1(1/4) &= (1/12, 1/24) = (0.0833, 0.0417) \\ P_2(1/4) &= (1/14, 1/28) = (0.0714, 0.0357) \\ P_3(1/4) &= (1/16, 1/32) = (0.0625, 0.0313) \\ P_4(1/4) &= (1/18, 1/36) = (0.0556, 0.0278). \end{aligned}$$

If we increase α , the points $P_1(\alpha)$ to $P_k(\alpha)$ will “slide down” along the line $\beta_1 = 2\beta_2$ with various speed. Let $j = 2, 3, \dots, k$. In the following, we compute the value of α such that $P_j(\alpha)$ and $P_{j-1}(\alpha)$ coincide. We have already shown in Lemma 6 that $\beta_1 = 2\beta_2$. Hence, it suffices to solve

$$\begin{aligned} B &= (k-j)\alpha + \left(j(d-k) + \frac{j^2+j}{2}\right)\beta_1 + j(r-1)\frac{\beta_1}{2} \\ B &= (k-j+1)\alpha + \left((j-1)(d-k) + \frac{(j-1)^2+j-1}{2}\right)\beta_1 \\ &\quad + (j-1)(r-1)\frac{\beta_1}{2}. \end{aligned}$$

By subtracting one of the above equations from the other, we obtain

$$\begin{aligned} \beta_1 &= \frac{2B}{k(2d-2k+2j+r-1) - j(j-1)} = \frac{2B}{D_j} \\ \alpha &= (2(d-k+j) + r-1)\frac{B}{D_j}. \end{aligned}$$

The corresponding repair-bandwidth γ is

$$\gamma = d\beta_1 + (r-1)\beta_2 = (2d+r-1)\frac{B}{D_j}.$$

This gives the operating point of the first type in (9). The constant $\tilde{\alpha}_j$ in (2) is defined such that $P_j(\tilde{\alpha}_j) = P_{j-1}(\tilde{\alpha}_j)$.

In Fig. 8, we note that the lines $L_1(1/4)$, $L_2(1/4)$, and $L_3(1/4)$, intersect at the same point on the line $\beta_1 = \beta_2$. The operating points of the second type are obtained by generalizing this observation. For notational convenience, we let $L_0(\alpha)$ be the set of points in the β_1 - β_2 plane satisfying the equation $B = \alpha k$, i.e., it is either the whole plane if $\alpha = B/k$ or the empty set if $\alpha > B/k$.

Lemma 8. *Let ℓ be an integer between 0 and $\lfloor k/r \rfloor$. We can choose α such that $L_j(\alpha)$, for $j = \ell r, \ell r + 1, \dots, \ell r + r$, and the line $\beta_1 = \beta_2$ have a common intersection point in the β_1 - β_2 plane.*

Proof: Let j be an integer between ℓr and $\ell r + r$. We can write $j = \ell r + c$ for some integer c in the range $0 \leq c \leq r$, and get

$$\Delta_{\ell r+c,r} = \ell r^2 + c^2.$$

The equation of $L_j(\alpha)$, for $\ell r \leq j \leq \ell r + r$, is

$$\begin{aligned} B &= (k - \ell r - c)\alpha + \left((\ell r + c)(d - k) \right. \\ &\quad \left. + \frac{\ell^2 r^2 + \ell r^2 + 2\ell r c + 2c^2}{2}\right)\beta_1 + c(r - c)\beta_2. \end{aligned}$$

Substitute $\beta_1 = \beta_2$ into the above equation, we can re-write it as

$$\begin{aligned} B &= c(-\alpha + [d - k + r(\ell + 1)]\beta_1) \\ &\quad + (k - \ell r)\alpha + \left[\ell r(d - k) + r^2\ell(\ell + 1)/2\right]\beta_1. \end{aligned}$$

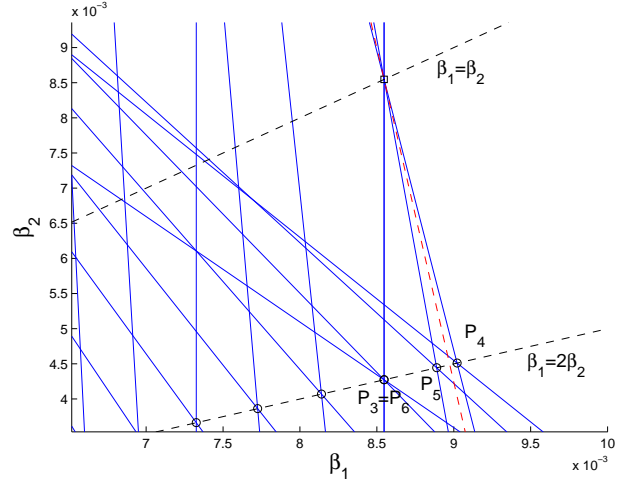


Fig. 9. The linear programming problem for $d = 19$, $k = 18$, $r = 3$, $B = 1$, and $\alpha = 7/117 = 0.0598$. The objective function $19\beta_1 + 2\beta_2$ is minimized at the point $Q_1 = (1/117, 1/117) = (0.00854, 0.00854)$.

When $\alpha = [d + r(\ell + 1) - k]\beta_1$, we can eliminate c and get

$$\begin{aligned} B &= \beta_1[(k - \ell r)(d - k + r(\ell + 1)) + \ell r(d - k) \\ &\quad + r^2\ell(\ell + 1)/2], \end{aligned}$$

which can be further simplified to

$$\beta_1 = \frac{B}{k(d + r(\ell + 1) - k) - r^2\ell(\ell + 1)/2} = \frac{B}{D'_\ell}.$$

Hence, when $\alpha = B(d + r(\ell + 1) - k)/D'_\ell$, the point $(B/D'_\ell, B/D'_\ell)$ in the β_1 - β_2 plane is an common intersection point of $L_j(\alpha)$, for $j = \ell r, \ell r + 1, \dots, \ell r + r$. ■

Definitions: For $\ell = 0, 1, 2, \dots, \lfloor k/r \rfloor$, define Q_ℓ as the point

$$Q_\ell := (B/D'_\ell, B/D'_\ell) \quad (33)$$

in the β_1 - β_2 plane.

We note that when $\ell = 0$, we have

$$Q_0 = (B/(k(d + r - k)), B/(k(d + r - k))).$$

An illustration is shown in Fig. 9. The point Q_1 is the point marked by a square on the line $\beta_1 = \beta_2$.

Remarks: In the special case of single-loss recovery, i.e., when $r = 1$, the variable β_2 can take any value without affecting the repair-bandwidth, because the second phase of repair is vacuous. The line $L_j(\alpha)$ and $L'_j(\alpha)$ representing the linear constraints are vertical lines in the β_1 - β_2 plane. Naturally, we take $\beta_2 = 0$ when $r = 1$. However, in order to give a unified treatment covering the two cases $r = 1$ and $r \geq 2$, we define $P_j(\alpha)$ and Q_ℓ for all $r \geq 1$, even though the β_2 coordinates of $P_j(\alpha)$ and Q_ℓ are nonzero. The results in the next two sections hold for both $r = 1$ and $r \geq 2$.

V. CONSTRUCTION OF MAXIMAL FLOW

To ease the presentation, we modify the information flow graph by adding more “out” vertices, so that in each stage, each storage node is associated with a unique “out” vertex. If the storage node is not repaired in stage s , we draw a directed

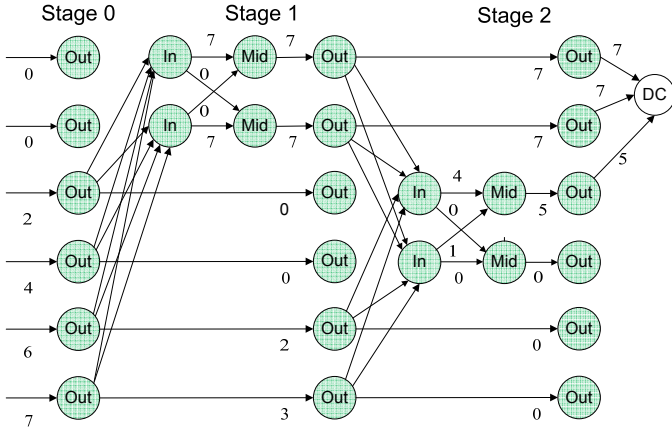


Fig. 10. An example of modified information flow graph ($n = 6$, $d = 4$, $k = 3$, $r = 2$, $\alpha = 7$, $\beta_1 = 2$, $\beta_2 = 1$).

edge with infinite capacity from the “out” node in stage $s - 1$ to it. With the addition of these new vertices, all inter-stage edges are between two consecutive stages. A modified information flow graph is denoted by $G^m(n, d, k, r; \alpha, \beta_1, \beta_2)$. The modified information flow graph $G^m(6, 4, 3, 2; 7, 2, 1)$ derived from the example in Fig. 5 is shown in Fig. 10.

We consider the cuts of the modified information flow graph which separate two consecutive stages. The flow pattern through such these “vertical” cuts are captured by the vectors defined in the following definition.

Definitions: Consider a data collector DC in stage s , $s \geq 0$. Given a particular flow F on G , for $t = 0, 1, 2, \dots, s - 1$, let $\mathbf{h}^{(t)} = [h_1^{(t)} h_2^{(t)} \dots h_n^{(t)}]$ be the n -dimensional vector whose i -th component is the sum of flow through the vertex Out_i in stage t . We call the vector $\mathbf{h}^{(t)}$ a *flow pattern*.

We will use superscript (t) to signify that a vector or variable is pertaining to stage t .

For example in Fig. 10, the in-flows of the six vertices in stage 0 are respectively 0, 0, 2, 4, 6, and 7. Hence

$$\mathbf{h}^{(0)} = [0 \ 0 \ 2 \ 4 \ 6 \ 7]. \quad (34)$$

The first two components are zero because node 1 and 2 fail in stage 0, and they do not have any out-flow. In stage 2, nodes 3 and 4 are repaired. We get

$$\mathbf{h}^{(1)} = [7 \ 7 \ 0 \ 0 \ 2 \ 3]. \quad (35)$$

We note if $i \in \mathcal{R}_t$, then node i fails in stage $t - 1$, and the i -th component in $\mathbf{h}^{(t-1)}$ is equal to zero. Finally, the three edges which terminate at the DC have flow 7, 7 and 5 respectively, and we get

$$\mathbf{h}^{(2)} = [7 \ 7 \ 5 \ 0 \ 0 \ 0]. \quad (36)$$

As a simple non-example, we note that any vector with a component strictly larger than α is not transmissive in any stage.

Definitions: A vector $\mathbf{v} \in \mathbb{R}_+^n$ is called *transmissive at stage s* ($s \geq 0$), if in any modified information flow graph $G^m(n, d, k, r; \alpha, \beta_1, \beta_2)$, we can assign flow $\phi(e)$ to the edges e in or before stage s , such that (i) $\phi(e)$ does not exceed the capacity of edge e , (ii) the sum of in-flow is equal to the sum of out-flow for all vertices in stage 1 to $s - 1$, and (iii) such that

the in-flow of the i -th “out” vertex in stage s is equal to the i th component in \mathbf{v} . A vector $\mathbf{v} \in \mathbb{R}_+^n$ which is transmissive at all stages is called *transmissive*.

Some comments on transmissive vector and flow pattern is in order. (i) A flow pattern is always attached with a data collector, but the definition of transmissive vector does not involve any data collector. (ii) a vector which is transmissive in one stage may not be transmissive in another stage. For example, in stage 0, the vector $[\alpha \ \alpha \ \dots \ \alpha]$ with all components equal to α is transmissive, but it is not a valid flow pattern (unless we are in the trivial case that $n = k$), and is not transmissive in stage 1.

For the operating points of the first type, we have the following theorem.

Theorem 9. Let z be an integer between 0 and $k - 2$, and let the parameters of the distributed storage system be

$$\alpha = 2(d - z) + r - 1, \text{ and} \\ \beta_1 = 2, \beta_2 = 1.$$

Then the max-flow to each DC is at least

$$B = k(2d + r - k) - z - z^2.$$

In fact, any vector $\mathbf{h} \in \mathbb{R}_+^n$ which is majorized by

$$[\underbrace{\alpha \ \dots \ \alpha}_{z \text{ times}} \ \underbrace{\alpha \ \alpha - 2 \ \alpha - 4 \ \dots \ \alpha - 2(k - z - 1)}_{k - z \text{ terms}} \ \underbrace{0 \ \dots \ 0}_{n - k \text{ times}}]. \quad (37)$$

are transmissive. Furthermore, if the components of \mathbf{h} are non-negative integers, then we can choose an integral flow in the modified information flow graph.

We can prove By Theorem 9 that the vector $\mathbf{h}^{(0)}$ in (34), $\mathbf{h}^{(1)}$ in (35), and $\mathbf{h}^{(2)}$ in (36) are all transmissive in the modified information flow graph in Fig. 10. Indeed, they are all majorized by $[7 \ 7 \ 5 \ 0 \ 0 \ 0]$. We can prove that they are transmissive by applying Theorem 9 with $z = 1$.

We check that the sum of the components in (37) is equal to B ,

$$\begin{aligned} k\alpha - \sum_{i=1}^{k-z-1} 2i \\ = k(2d - 2z + r - 1) - (k - z - 1)(k - z) \\ = k(2d + r - k) - z - z^2 = B. \end{aligned}$$

Let \mathcal{P} be the set of vectors in \mathbb{R}_+^n which is majorized by the vector in (37). By Lemma 2, the set of vectors \mathcal{P} is a base-polymatroid

$$\{\mathbf{x} \in \mathbb{R}_+^n : \mathbf{x}(S) \leq f(S) \text{ for } S \subseteq \{1, 2, \dots, n\} \\ \text{and } \sum_{i=1}^n x_i = B\}$$

associated with the rank function $f : \{0, 1, 2, \dots, n\} \mapsto \mathbb{Z}_+$ given by $f(S) = \theta_{|S|}$, where θ_j is defined by

$$\theta_j := \min(k, j) \cdot \alpha - \sum_{i=0}^{\min(k, j) - z - 1} 2i \quad (38)$$

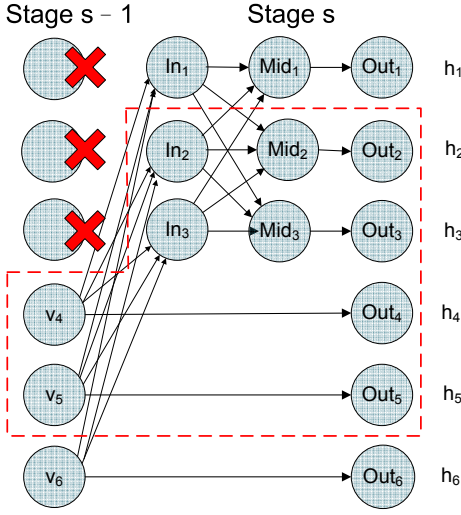


Fig. 11. An example of the auxiliary graph.

for $j = 0, 1, 2, \dots, n$. (If the upper limit of a summation is negative, the summation is equal to 0 by convention.) We check that θ_x is equal to the sum of the first x terms in (37), and $\theta_k = \theta_{k+1} = \dots = \theta_n = B$.

The proof of Theorem 9 relies on the trellis structure of the modified information flow graph. The subgraph obtained by restricting to one stage are isomorphic to the subgraph obtained by restricting to another stage. This allows us to simplify the analysis to only one stage. Consider the subgraph of the modified information flow graph consisting of the vertices in stage s and the n “out” vertices in stage $s-1$. We call this the *auxiliary graph*, and let \mathcal{V}' be the vertex set of this auxiliary graph. By re-labeling the storage nodes, we assume without loss of generality that nodes 1 to r are regenerated in stage s . The first r “out” vertices in stage $s-1$ is disconnected from the rest of the auxiliary graph. In order to distinguish the “out” vertices in stage $s-1$ and s , we re-label the $n-r$ “out” vertices in stage $s-1$ by $v_{r+1}, v_{r+2}, \dots, v_n$. An example for $n = 6$ and $d = r = 3$ is given in Fig. 11.

The construction of flow in Theorem 9 is recursive. We consider the vertices on the left-hand side of the auxiliary graph as input vertices and the vertices on the right as output vertices. Let \mathbf{h} be a vector majorized by the vector in (37). The vector \mathbf{h} is regarded as the demand from the “out” nodes in the auxiliary graph. We look for a valid flow assignment in the auxiliary graph such that the flow to each “out” vertices is equal to the corresponding components in \mathbf{h} , and meanwhile the input flow assignment is majorized by (37).

We now specify a submodular function $\sigma : 2^{\mathcal{V}'} \rightarrow \mathbb{R}_+$. Let \mathcal{O}_{s-1} be the set of “out” vertices $\{v_{r+1}, v_{r+2}, \dots, v_n\}$ in stage $s-1$, and \mathcal{O}_s be the set of “out” vertices $\{\text{Out}_1, \text{Out}_2, \dots, \text{Out}_n\}$ in stage s . Given a subset \mathcal{S} of vertices in the auxiliary graph, define

$$\sigma(\mathcal{S}) := f(\mathcal{S} \cap \mathcal{O}_{s-1}) - \mathbf{h}(\mathcal{S} \cap \mathcal{O}_s).$$

The notation $\mathbf{h}(\mathcal{S} \cap \mathcal{O}_s)$ in the above definition means

$$\mathbf{h}(\mathcal{S} \cap \mathcal{O}_s) = \sum_{\text{Out}_i \in \mathcal{S}} h_i.$$

The function $\sigma(\mathcal{S})$ is submodular because it is the sum of a submodular function $f(\mathcal{S} \cap \mathcal{O}_{s-1})$ and a modular function $-\mathbf{h}(\mathcal{S} \cap \mathcal{O}_s)$. Also, we note that

$$\sigma(\mathcal{V}') = f(\mathcal{O}_{s-1}) - \mathbf{h}(\mathcal{O}_s) = B - B = 0.$$

We define upper bounds and lower bounds on the edges in the auxiliary graph as follows. For $i = r+1, r+2, \dots, n$, the edge joining v_i and Out_i has lower bound and upper bound equal to h_i . An edge terminating at an “in” vertex has lower bound 0 and upper bound β_1 . An edge from In_i to Mid_j for $i \neq j$, has lower bound 0 and upper bound β_2 , while an edge from In_i to Mid_j for $i = j$, has lower bound 0 and upper bound ∞ . An edge from a “mid” vertex to an “out” vertex has lower bound 0 and upper bound α . We summarize the lower and upper bounds on the edges in the auxiliary graph as follows.

Edge e	$\alpha(e)$	$\beta(e)$
(v_i, Out_i)	h_i	h_i
(v_i, In_j)	0	β_1
$(\text{In}_i, \text{Mid}_j), i \neq j$	0	β_2
$(\text{In}_i, \text{Mid}_j), i = j$	0	∞
$(\text{Mid}_i, \text{Out}_i)$	0	α

To apply Theorem 3, we need to verify that condition (21) holds for all subsets $\mathcal{S} \subseteq \mathcal{V}'$.

Lemma 10. *With notation as in Theorem 9, we have*

$$\text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) \leq \rho(\mathcal{S}), \quad (39)$$

for all $\mathcal{S} \subseteq \mathcal{V}'$.

The proof of Lemma 10 is given in Appendix B.

Proof of Theorem 9: We proceed by induction on stages. Let \mathbf{h} be a vector in \mathcal{P} . Since each component of \mathbf{h} is less than or equal to α , we can always assign flow on the edges from the source vertex to the vertices in stage 0 such that $\mathbf{h}^{(0)} = \mathbf{h}$, without violating any capacity constraint. Hence \mathbf{h} is transmissive at stage 0.

Suppose that all vectors in \mathcal{P} are transmissive in stage $s-1$. Consider the auxiliary graph consisting of the vertices in stage s and the n “out” vertices in stage $s-1$. By applying Frank’s theorem (Theorem 3), there exists a feasible submodular flow, say ϕ , on the auxiliary graph. The flow conservation constraint is satisfied for the “in” and “mid” vertices in the auxiliary graph, because by definition $\sigma(\text{In}_i) = \sigma(\text{Mid}_i) = 0$. Since the submodular flow guaranteed by Frank’s theorem is feasible, the capacity constraints in the modified information flow graph are satisfied.

If we take any subset \mathcal{A} of the “out” vertices in stage $s-1$, from the definition of a submodular flow, we obtain

$$\partial\phi(\mathcal{A}) = \phi(\Delta_{\mathcal{A}}^{\text{out}}) \leq \sigma(\mathcal{A}) = f(\mathcal{A}).$$

The “input” in the $(s-1)$ -st stage is thus transmissive at stage $s-1$. By the induction hypothesis, we can assign real values to the edges from stage -1 to $s-1$ in the modified information flow graph, such that the flow conservation constraint is satisfied, and the in-flow of the “out” vertices Out_i in stage $s-1$ is precisely the inputs of the corresponding vertices in the auxiliary graph.

Finally, we want to show that

$$\phi(\Delta_{\{\text{Out}_i\}}^{\text{in}}) = -h_i$$

for $i = 1, 2, \dots, n$. By the defining property of a submodular flow again, we have

$$\partial\phi(\{\text{Out}_i\}) = -\phi(\Delta_{\{\text{Out}_i\}}^{\text{in}}) \leq -h_i.$$

If we take \mathcal{S} be the subsets

$$\mathcal{S} = \{\text{In}_1, \text{In}_2, \dots, \text{In}_r, \text{Mid}_1, \text{Mid}_2, \dots, \text{Mid}_r\}$$

we have

$$\begin{aligned} 0 = \sigma(\mathcal{S}) &\geq \partial\phi(\mathcal{S}) = \phi(\Delta_{\mathcal{S}}^{\text{out}}) - \phi(\Delta_{\mathcal{S}}^{\text{in}}) \\ &= \sum_{i=1}^n \phi(\Delta_{\{\text{Out}_i\}}^{\text{in}}) - \sum_{i=1}^n \phi(\Delta_{\{\text{In}_i\}}^{\text{out}}) \\ &\geq \sum_{i=1}^n h_i - \sigma(\{v_1, v_2, \dots, v_n\}) = B - B = 0. \end{aligned}$$

Therefore, all inequalities above are in fact equalities. Thus $\phi(\Delta_{\{\text{Out}_i\}}^{\text{in}}) = h_i$ for all i . This gives a flow on the s -th stage of the modified information flow graph yielding the desired vector \mathbf{h} . This proves that \mathbf{h} is transmissive at stage s .

If the components of \mathbf{h} are non-negative integers, then the result follow from the second statement of Frank's theorem. This completes the proof of Theorem 9. ■

Theorem 11. For $j = 2, 3, \dots, k$, the operating point $(\tilde{\gamma}_j, \tilde{\alpha}_j)$ (defined in (2) and (3)) is in $\mathcal{C}_{\text{MF}}(d, k, r)$.

Proof: Consider a data collector DC who connects two k storage nodes in stage s . We want to construct a flow from the source node to DC such that the flow of the k links from the k storage nodes to the data collector be precisely the non-zero components in (37). The require flow pattern is certainly majorized by (37). By Theorem 9, we can always find a flow meeting the requirement of this data collector, regardless of which storage nodes failed in earlier stages. Hence, for $z = 0, 1, \dots, k-2$, the operating point

$$\frac{1}{k(2d+r-k)-z-z^2}(2d+r-1, 2(d-z)+r-1)$$

is in $\mathcal{C}_{\text{MF}}(d, k, r)$. After a change of the indexing variable by $k = z + j$, we see that, for $j = 2, 3, \dots, k$,

$$(\tilde{\gamma}_j, \tilde{\alpha}_j) = \frac{1}{D_j}(2d+r-1, 2(d-k+j)+r-1)$$

is in $\mathcal{C}_{\text{MF}}(d, k, r)$. ■

Analogous to Theorem 9 and Theorem 11, we have the following two theorems.

Theorem 12. Let ℓ be an integer between 0 and $\lfloor k/r \rfloor$, and let the parameters of a distributed storage system be

$$\begin{aligned} \alpha &= d + r(\ell + 1) - k, \text{ and} \\ \beta_1 &= \beta_2 = 1. \end{aligned}$$

The max-flow to each DC is at least

$$B = k(d + r(\ell + 1) - k) - r^2\ell(\ell + 1)/2.$$

In fact, any vector $\mathbf{h} \in \mathbb{R}_+^n$ majorized by

$$\begin{aligned} &\underbrace{[\alpha \dots \alpha]}_{k-\ell r \text{ times}} \underbrace{[\alpha-r \dots \alpha-r]}_{r \text{ times}} \underbrace{[\alpha-2r \dots \alpha-2r]}_{r \text{ times}} \\ &\dots \underbrace{[\alpha-\ell r \dots \alpha-\ell r]}_{r \text{ times}} \underbrace{[0 \dots 0]}_{n-k \text{ times}} \end{aligned} \quad (40)$$

is transmissive. Furthermore, if the components of \mathbf{h} are non-negative integers, then we can choose an integral flow in the modified information flow graph.

Theorem 13. For $\ell = 0, 1, 2, \dots, \lfloor k/r \rfloor$, the operating point $(\tilde{\gamma}'_\ell, \tilde{\alpha}'_\ell)$ (defined in (5) and (6)) is in $\mathcal{C}_{\text{MF}}(d, k, r)$.

The proof of Theorem 12 is given in Appendix C.

In summary, we have shown in Theorems 11 and 13 that

$$\gamma_{\text{MF}}^*(\tilde{\alpha}_j) \leq \tilde{\gamma}_j,$$

for $j = 2, 3, \dots, k$, and

$$\gamma_{\text{MF}}^*(\tilde{\alpha}'_\ell) \leq \tilde{\gamma}'_\ell,$$

for $\ell = 0, 1, \dots, \lfloor k/r \rfloor$. Next, we show that under the conditions in (9) and (10), these are indeed the optimal value of $\gamma_{\text{MF}}^*(\tilde{\alpha}_j)$ or $\gamma_{\text{MF}}^*(\tilde{\alpha}'_\ell)$.

Theorem 14. If $d \leq (r-1)\mu(j)$ for any $j = 2, 3, \dots, k$, then

$$\gamma_{\text{LP}}^*(\tilde{\alpha}_j) = \gamma_{\text{MF}}^*(\tilde{\alpha}_j) = \tilde{\gamma}_j.$$

If $d > (r-1)\mu(j)$ for any $j = 0, 1, 2, \dots, k$, then

$$\gamma_{\text{LP}}^*(\tilde{\alpha}'_{\lfloor j/r \rfloor}) = \gamma_{\text{MF}}^*(\tilde{\alpha}'_{\lfloor j/r \rfloor}) = \tilde{\gamma}'_{\lfloor j/r \rfloor}.$$

Proof: The proof is based on the following fact in linear programming with two variables. Suppose the slope of the objective function is m , and \mathbf{x} is a feasible solution. If \mathbf{x} satisfies two linear constraints, one with slope larger than or equal to m and one with slope smaller than or equal to m , with equality, then \mathbf{x} is the optimal solution. Recall that the objective function of the linear program considered in this paper has slope $-d/(r-1)$, which is ∞ when $r = 1$.

For the first statement in the theorem, the point $P_j(\tilde{\alpha}_j)$ in the β_1 - β_2 plane is a feasible solution by Theorem 11, and meets the two linear constraints corresponding to $L_j(\tilde{\alpha}_j)$ and $L'_j(\tilde{\alpha}_j)$ with equality. The optimality of $P_j(\tilde{\alpha}_j)$ follows from (i) line $L_j(\tilde{\alpha}_j)$ has magnitude $\mu(j)$, which is larger than or equal to the magnitude of the objective function by hypothesis, (ii) line $L'_j(\tilde{\alpha}_j)$ has magnitude less than $d/(r-1)$ (See Lemma 6 part (2)). Therefore $P_j(\tilde{\alpha}_j)$ is the optimal solution. The corresponding α and γ are $\tilde{\alpha}_j$ and $\tilde{\gamma}_j$ respectively. Therefore $\gamma_{\text{MF}}^*(\tilde{\alpha}_j) = \tilde{\gamma}_j$.

For the second statement in the theorem, the point Q_ℓ in the β_1 - β_2 plane is a feasible solution by Theorem 13. The point Q_ℓ meets two linear constraints with equality. The first one is the inequality associated with line $L_{\lfloor j/r \rfloor}(\tilde{\alpha}'_{\lfloor j/r \rfloor})$, which is a vertical line. The second one is the inequality associated with line $L_j(\tilde{\alpha}'_{\lfloor j/r \rfloor})$, whose slope has magnitude strictly less than $d/(r-1)$ by hypothesis. Therefore Q_ℓ is the optimal solution to the linear program. The corresponding α and γ are $\tilde{\alpha}'_{\lfloor j/r \rfloor}$ and $\tilde{\gamma}'_{\lfloor j/r \rfloor}$ respectively. Therefore $\gamma_{\text{MF}}^*(\tilde{\alpha}'_{\lfloor j/r \rfloor}) = \tilde{\gamma}'_{\lfloor j/r \rfloor}$. ■

We check that the MSCR and MBCR point are boundary points in \mathcal{C}_{MF} . Firstly, the slope of the line $L_1(\alpha)$ has magnitude

$$\frac{d-k+1}{r-1},$$

which is strictly less than $d/(r-1)$. We get

$$\gamma_{\text{LP}}^*(\tilde{\alpha}_{\text{MSCR}}) = \gamma_{\text{MF}}^*(\tilde{\alpha}_{\text{MSCR}}) = \tilde{\gamma}'_0 = \tilde{\gamma}_{\text{MSCR}}.$$

Secondly, the condition $d \leq (r-1)\mu(k)$ reduces to

$$d \leq (r-1) \frac{d + \Delta_{k,r}/(2k)}{r - \Delta_{k,r}/k},$$

which holds for all positive integers d, k and r . Hence

$$\gamma_{\text{LP}}^*(\tilde{\alpha}_{\text{MBCR}}) = \gamma_{\text{MF}}^*(\tilde{\alpha}_{\text{MBCR}}) = \tilde{\gamma}_k = \tilde{\gamma}_{\text{MBCR}}.$$

Consequently, we have shown that the operating points in Theorem 1 are boundary points in \mathcal{C}_{LP} and \mathcal{C}_{MF} . The next theorem shows that there is no gap between \mathcal{C}_{LP} and \mathcal{C}_{MF} .

Theorem 15. $\mathcal{C}_{\text{LP}} = \mathcal{C}_{\text{MF}}$.

The proof of Theorem 15 is technical and is given in Appendix D.

VI. LINEAR NETWORK CODES FOR COOPERATIVE REPAIR

In this section, we show that the Pareto-optimal operating points in \mathcal{C}_{MF} can be achieved by linear network coding, with an explicit bound on the required finite field size.

Let \mathbb{F}_q denote the finite field of size q , where q is a power of prime. The size of \mathbb{F}_q will be determined later in this section. We normalize the unit such that an element in \mathbb{F}_q contains one unit of data. The whole data file is divided into a number of chunks, and each chunk contains B finite field elements. As each chunk of data will be encoded and treated in the same way, it suffices to describe the operations on one chunk of data. A packet is identified with an element in \mathbb{F}_q , and we will use “an element in \mathbb{F}_q ”, “a packet” and “a symbol” synonymously.

We scale the value of B, β_1, β_2 , and α , so that they are all integers. A chunk of data is represented by a B -dimensional column vector $\mathbf{m} \in \mathbb{F}_q^B$. The data packet stored in a storage node is a linear combination of the components in \mathbf{m} , with coefficients taken from \mathbb{F}_q . The coefficients associated with a packet form a vector, called the *global encoding vector*. For $i = 1, 2, \dots, n$, and $t \geq 0$, the packets stored in node i are denoted by the vector $\mathbf{M}_i^{(t)} \mathbf{m}$, where $\mathbf{M}_i^{(t)}$ is an $\alpha \times B$ matrix. The rows of $\mathbf{M}_i^{(t)}$ are the global encoding vectors of the packets in node i in stage t . We will assume that the global encoding vectors are stored together with the packets in the storage nodes. The overhead on storage incurred by the global encoding vectors can be made vanishingly small if the number of chunks is large. The (n, k) recovery property is translated to the requirement that the totality of the global encoding vectors in any k storage nodes span the vector space \mathbb{F}_q^B .

The code construction method we are going to describe maintains the following property, which we will call the *regularity property*. In the followings, \mathcal{P} is a set of transmissive vectors with integral components, such that the sum of the components of each vector in \mathcal{P} is equal to B .

Regularity Property with respect to \mathcal{P} : For $t \geq 0$ and for each vector $\mathbf{h} = [h_1 \ h_2 \ \dots \ h_n]$ in \mathcal{P} , if we take the first h_i rows of $\mathbf{M}_i^{(t)}$ for each i , and putting them together as a $B \times B$ matrix, then the resulting determinant, denoted by $D_{\mathbf{h}}^{(t)}$, is non-zero.

The realization of cooperative repair using linear network coding is described as follows.

Stage 0: For $i = 1, 2, \dots, n$, node i is initialized by storing the α components in $\mathbf{M}_i^{(0)} \mathbf{m}$.

Stage t : For notational convenience, we suppose without loss of generality that node 1 to node r fail at stage t , and we want to regenerate them in stage $t+1$.

- *Phase 1.* For $j = 1, 2, \dots, r$ and $i \in \mathcal{H}_{t,j}$, the β_1 packets sent from node i to node j are linear combinations of the packets stored in node i at stage t . For $\ell = 1, 2, \dots, \beta_1$, let the ℓ -th packet sent from node i to node j be $\mathbf{p}_{ij\ell}^{(t)} \mathbf{M}_i^{(t)} \mathbf{m}$, where $\mathbf{p}_{ij\ell}^{(t)}$ is a $1 \times \alpha$ row vector over \mathbb{F}_q .
- *Phase 2.* Stack the $d\beta_1$ received packets by node j into a column vector called $\mathbf{u}_j^{(t)}$. For $j_1, j_2 \in \{1, 2, \dots, r\}$ and $j_1 \neq j_2$, node j_1 sends β_2 packets to node j_2 . For $\ell = 1, 2, \dots, \beta_2$, the ℓ -th packet sent from node j_1 to node j_2 is $\mathbf{q}_{j_1,j_2,\ell}^{(t)} \mathbf{u}_{j_1}^{(t)}$, where $\mathbf{q}_{j_1,j_2,\ell}^{(t)}$ is a $(d\beta_1)$ -dimensional row vector over \mathbb{F}_q .

The $(r-1)\beta_2$ packets received by newcomer j during phase 2 are put together to form an $((r-1)\beta_2)$ -dimensional column vector $\mathbf{v}_j^{(t)}$. For $\ell = 1, 2, \dots, \alpha$, newcomer j multiplies a $(d\beta_1 + (r-1)\beta_2)$ -dimensional row vector $\mathbf{r}_j^{(t)}$ with the column vector obtained by concatenating $\mathbf{u}_j^{(t)}$ and $\mathbf{v}_j^{(t)}$, and stores the product as the ℓ -th packet in the memory.

The vector $\mathbf{p}_{ij\ell}^{(t)}$'s, $\mathbf{q}_{j_1,j_2,\ell}^{(t)}$'s and $\mathbf{r}_j^{(t)}$'s are called the *local encoding vectors*. The components in the local encoding vectors are variables assuming value in \mathbb{F}_q . The total number of “degrees of freedom” in choosing the local encoding vectors is

$$N = rd\beta_1\alpha + r(r-1)\beta_2(d\beta_1) + r\alpha(d\beta_1 + (r-1)\beta_2).$$

We will call these N variables the *local encoding kernels* at stage t . We will show that, given the global encoding vectors of the n nodes in stage $t-1$, we can find local encoding kernels in stage t such that the (n, k) recovery property is maintained, provided that the finite field size is sufficiently large.

The proof depends on the trellis structure of the modified information flow graph defined in the last section. The “transfer function” can be factorized as a products of matrices.

We concatenate all packets in the n storage nodes in stage t into an $(n\alpha)$ -dimensional vector, and write

$$\mathbf{s}^{(t)} := \mathbf{M}^{(t)} \mathbf{m},$$

where $\mathbf{M}^{(t)}$ is the $(\alpha n) \times B$ matrix

$$\mathbf{M}^{(t)} := \begin{bmatrix} \mathbf{M}_1^{(t)} \\ \mathbf{M}_2^{(t)} \\ \vdots \\ \mathbf{M}_n^{(t)} \end{bmatrix}.$$

The packets in stage t can be obtained by multiplying $\mathbf{s}^{(t-1)}$ by an $(n\alpha) \times (n\alpha)$ matrix $\mathbf{T}^{(t)}$,

$$\mathbf{s}^{(t)} = \mathbf{T}^{(t)} \mathbf{s}^{(t-1)}. \quad (41)$$

If nodes 1 to r fails, the matrix $\mathbf{T}^{(t)}$ can be partitioned into

$$\mathbf{T}^{(t)} = \left[\begin{array}{c|c} \mathbf{0} & \mathbf{A} \\ \hline \mathbf{0} & \mathbf{I} \end{array} \right],$$

where \mathbf{I} is the identity matrix of size $(n-r)\alpha \times (n-r)\alpha$, and \mathbf{A} is an $r\alpha \times (n-r)\alpha$ matrix. The entries of \mathbf{A} are polynomials with the N local encoding kernels at stage t as the variables. Moreover, the entries in \mathbf{A} have degree 1 in each variable. We can see this by fixing all but one variables in the local encoding kernels, and then the packets stored in the r newcomers' memory are affine functions of the remaining variable.

Applying (41) recursively, we obtain

$$\mathbf{s}^{(t)} = \mathbf{T}^{(t)} \mathbf{T}^{(t-1)} \dots \mathbf{T}^{(0)} \mathbf{m},$$

for $t \geq 0$. The matrix $\mathbf{T}^{(0)}$ is an $(n\alpha) \times B$ rectangular matrix. Each entries in $\mathbf{T}^{(0)}$ is a variable taking value in \mathbb{F}_q . Using this representation, we see that the rows in $\mathbf{M}_i^{(t)}$ are rows $(i-1)\alpha + 1, (i-1)\alpha + 2, \dots, i\alpha - 1$ in the product $\mathbf{T}^{(t)} \mathbf{T}^{(t-1)} \dots \mathbf{T}^{(0)}$.

We will need the following tool due to N. Alon. By a non-zero mutli-variable polynomial, we mean a polynomial, when expressed as a summation of monomials, that has at least one term with non-zero coefficient.

Lemma 16 (Combinatorial Nullstellensatz [32]). *Let $f(x_1, x_2, \dots, x_N)$ be a non-zero multi-variable polynomial over \mathbb{F}_q of total degree D , which contains a non-zero coefficient at $x_1^{D_1} x_2^{D_2} \dots x_N^{D_N}$ with $D_1 + D_2 + \dots + D_N = D$. Let $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_N$ be subsets of \mathbb{F}_q such that $|\mathcal{S}_i| > D_i$ for all i . Then there exist $a_1 \in \mathcal{S}_1, \dots, a_N \in \mathcal{S}_N$ such that $f(a_1, \dots, a_N) \neq 0$.*

We say that the *local degree* of a polynomial f is less than or equal to ℓ if the degree of f in each variable is less than or equal to ℓ . The Combinatorial Nullstellensatz directly implies that if f is a non-zero polynomial in $\mathbb{F}_q[x_1, x_2, \dots, x_N]$ with local degree ℓ , then there is a point $(a_1, a_2, \dots, a_N) \in \mathbb{F}_q^N$ such that $f(a_1, a_2, \dots, a_N) \neq 0$.

We treat the two different types of Pareto-optimal operating points, one described in Theorem 9 and one in Theorem 12, separately.

Pareto-optimal operating point of the first type: We let the system parameters be the same as in Theorem 11. Let j be an integer between 2. Consider a linear cooperative regenerating code with the following parameters:

$$\begin{aligned} B &= k(2d + r - k) - (k - j)(k - j + 1), \\ \beta_1 &= 2, \quad \beta_2 = 1, \\ \alpha &= 2(d - k + j) + r - 1, \text{ and} \\ \gamma &= 2d + r - 1. \end{aligned}$$

Let \mathcal{P}_1 be the subset of vectors in \mathbb{Z}_+^n which is majorized by

$$\underbrace{[\alpha \dots \alpha]_{k-j \text{ times}}} \underbrace{[\alpha \quad \alpha - 2 \quad \alpha - 4 \quad \dots \quad \alpha - 2(j-1)]_j \text{ terms}} \underbrace{[0 \dots 0]_{n-k \text{ times}}}.$$

Let $|\mathcal{P}_j|$ be the cardinality of \mathcal{P}_j .

To initialize the system, we choose the entries in $\mathbf{T}^{(0)}$ such that the regularity property with respect to \mathcal{P}_j holds at stage 0, i.e., the determinant $D_{\mathbf{h}}^{(0)}$ defined in the regularity property is non-zero for all $\mathbf{h} \in \mathcal{P}_j$. This is equivalent to choose $\mathbf{T}^{(0)}$ such that $\prod_{\mathbf{h} \in \mathcal{P}_j} D_{\mathbf{h}}^{(0)} \neq 0$. Recall that $D_{\mathbf{h}}^{(0)}$ is a $B \times B$ matrix whose entries are B^2 distinct variables. We can loosely upper bound the local degree of $\prod_{\mathbf{h} \in \mathcal{P}_j} D_{\mathbf{h}}^{(0)}$ by $|\mathcal{P}_j|$. By Combinatorial Nullstellensatz, we can choose $\mathbf{T}^{(0)}$ such that the regularity property is satisfied at $t = 0$ if $q > |\mathcal{P}_j|$.

Suppose that $D_{\mathbf{h}}^{(t-1)}$ is non-zero for all $\mathbf{h} \in \mathcal{P}_j$. For each $\mathbf{h} \in \mathcal{P}_j$, we let $\mathbf{T}_{\mathbf{h}}^{(t)}$ be the $B \times (n\alpha)$ submatrix of $\mathbf{T}^{(t)}$ obtained by extracting the rows associated with \mathbf{h} . If the rows of $\mathbf{T}^{(t)}$ is divided into n blocks, with each block consisting of α rows, then $\mathbf{T}_{\mathbf{h}}^{(t)}$ is obtained by retaining the first h_i rows of the i -th block of rows of $\mathbf{T}_{\mathbf{h}}^{(t)}$, for $i = 1, 2, \dots, n$.

The determinant $D_{\mathbf{h}}^{(t)}$ can be written as

$$D_{\mathbf{h}}^{(t)} = \det(\mathbf{T}_{\mathbf{h}}^{(t)} \mathbf{M}^{(t-1)}).$$

The entries in $\mathbf{T}_{\mathbf{h}}^{(t)}$ involve the local encoding kernels to be determined, but the entries in $\mathbf{M}^{(t-1)}$ are fixed elements in \mathbb{F}_q . By Theorem 9 (identifying z with $k - j$), there is an integral flow in the auxiliary graph with input \mathbf{g} and output \mathbf{h} , where \mathbf{g} is an integral transmissive vector. This means that if the local encoding kernels are chosen appropriately, the square submatrix of $\mathbf{T}_{\mathbf{h}}^{(t)}$ obtained by retaining the columns associated with \mathbf{g} is a permutation of the identity matrix, while the other columns not associated with \mathbf{g} are zero. The square submatrix of $\mathbf{M}^{(t-1)}$ obtained by retaining the rows associated with \mathbf{g} has non-zero determinant by the induction hypothesis. We can thus choose the local encoding kernels such that $D_{\mathbf{h}}^{(t)}$ is evaluated to a non-zero value. In particular, $D_{\mathbf{h}}^{(t)}$ is a non-zero polynomial with the local encoding kernels as the variables. After multiplying $D_{\mathbf{h}}^{(t)}$ over all $\mathbf{h} \in \mathcal{P}_j$, we see that $\prod_{\mathbf{h} \in \mathcal{P}_j} D_{\mathbf{h}}^{(t)}$ is also a non-zero polynomial.

Each local encoding kernel appears in at most $r\alpha$ rows in the determinant $D_{\mathbf{h}}^{(t)}$. The local degree of $\prod_{\mathbf{h} \in \mathcal{P}_j} D_{\mathbf{h}}^{(t)}$ can be upper bounded by $r\alpha|\mathcal{P}_j|$. By the Combinatorial Nullstellensatz, we can choose the local encoding vector in stage t such that the regularity property will continue to hold in stage t provided that

$$q > r\alpha|\mathcal{P}_j| = r(2(d - k + j) + r - 1)|\mathcal{P}_j|.$$

This proves that the Pareto-optimal operating point of the first type can be achieved by linear network coding.

Pareto-optimal operating point of the second type: Let i be an integer between 0 and $\lfloor k/r \rfloor$, and set

$$\begin{aligned} B &= k(d + r(i + 1) - k) - r^2 i(i + 1)/2, \\ \beta_1 &= \beta_2 = 1, \\ \alpha &= d - k + r(i + 1), \text{ and} \\ \gamma &= d + r - 1. \end{aligned}$$

Let \mathcal{Q}_i be the subset of vectors in \mathbb{Z}_+^n which is majorized

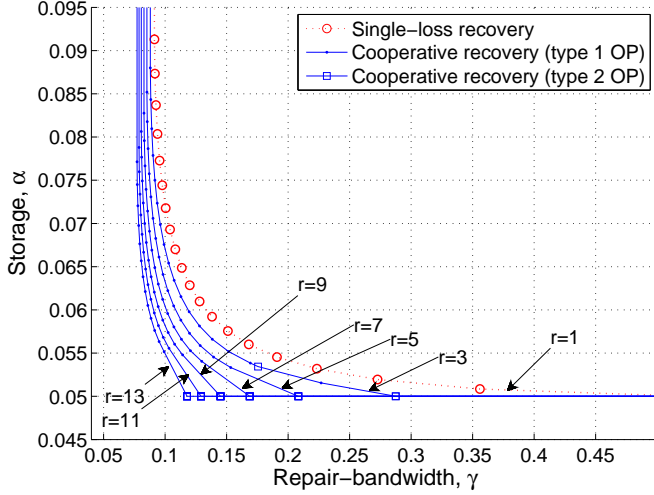


Fig. 12. Tradeoff between storage and repair-bandwidth ($d = 21$, $k = 20$, $B = 1$).

by

$$\begin{aligned} & \underbrace{[\alpha \dots \alpha]}_{k-ir \text{ times}} \underbrace{[\alpha - r \dots \alpha - r]}_{r \text{ times}} \underbrace{[\alpha - 2r \dots \alpha - 2r]}_{r \text{ times}} \\ & \dots \underbrace{[\alpha - ir \dots \alpha - ir]}_{r \text{ times}} \underbrace{[0 \dots 0]}_{n-k \text{ times}}. \end{aligned}$$

By similar arguments as in the previous subsection, we can guarantee that the regularity property with respect to Q_i is satisfied in all stages provided that $q > r(d - k + r(i + 1))|Q_i|$.

Theorem 17. *If the size of the finite field q is larger than*

$$\begin{aligned} & \max_{j=2, \dots, k} r(2(d - k + j) + r - 1)|P_j|, \text{ and} \\ & \max_{i=0, \dots, \lfloor k/r \rfloor} r(d - k + r(i + 1))|Q_i|, \end{aligned}$$

then we can implement linear network codes over \mathbb{F}_q for functional and cooperative repair, attaining the boundary points of \mathcal{C}_{MF} . Thus,

$$\mathcal{C}_{MF} = \mathcal{C}_{AD}.$$

Proof: We have already shown that the corner points of \mathcal{C}_{MF} can be achieved by linear network coding. By an analog of “time-sharing” argument, we see that all boundary points of \mathcal{C}_{MF} are achievable by linear network coding, if the finite field size is sufficiently large. Therefore, $\mathcal{C}_{AD} \subseteq \mathcal{C}_{MF}$. Since any operating point not in \mathcal{C}_{MF} are not admissible (see (17)), we conclude that $\mathcal{C}_{MF} = \mathcal{C}_{AD}$. ■

Remark: The requirement on finite field size in the previous theorem does not depend on the number of data collectors, and does not depend on the number of stages.

We have the following as an immediate corollary.

Corollary 18. *Given parameters d , k and r and B , the MSCR point $(\tilde{\gamma}_{MSCR}, \tilde{\alpha}_{MSCR})$ is achieved by $\beta_1 = \beta_2$. The MBCR point $(\tilde{\gamma}_{MBCR}, \tilde{\alpha}_{MBCR})$ is achieved by $\beta_1 = 2\beta_2$.*

In Fig. 12, we plot the tradeoff curves for distributed storage system with parameters $B = 1$, $d = 21$, $k = 20$, and $r =$

1, 3, 5, 7, 9, 11, 13. The number of storage nodes can be any integer larger than or equal to $d + 13 = 34$. The repair degree d is kept constant in the comparison. The curve for $r = 1$ is the tradeoff curve for single-node-repair regenerating code. Naturally, we have a better tradeoff curve when the number of cooperating newcomers increases. In Fig. 12, we indicate the Pareto-optimal operating points of the first type by dots and operating points of the second type by squares. With one exception, all Pareto-optimal points of the second type are the MSCR points.

We compare below the repair-bandwidth of three different modes of repair with minimum storage per node. The parameters are $n = 7$, $B = 1$, $k = 3$, and $\alpha = 1/3$. Suppose that three nodes have failed.

(i) Individual repair without newcomer cooperation. Each newcomer connects to the four remaining storage nodes. The repair-bandwidth per newcomer is

$$\frac{Bd}{k(d + 1 - k)} = \frac{4}{3(4 + 1 - 3)} = 0.6666.$$

(ii) One-by-one repair utilizing the newly regenerated node as a helper. The average repair-bandwidth per newcomer is

$$\frac{1}{3} \left(\frac{4}{3(4 + 1 - 3)} + \frac{5}{3(5 + 1 - 3)} + \frac{6}{3(6 + 1 - 3)} \right) = 0.5741.$$

The first term in the parenthesis is the repair-bandwidth of the first newcomer, which downloads from the four surviving nodes, the second term is the repair-bandwidth of the second newcomer, who connects to the four surviving nodes and the newly regenerated newcomer, and so on.

(iii) Full cooperation among the three newcomers. With $r = 3$, Corollary 18 gives the following lower bound in repair bandwidth,

$$\frac{4 + 3 - 1}{3(4 + 3 - 3)} = 0.5.$$

Finally, we compare the storage efficiency for fixed n , d and k , by increasing the number of nodes in cooperative repair. For MSCR, the storage efficiency is k/n , independent of r . For MBCR, the storage efficiency is

$$\frac{k(2d + r - k)}{n(2d + r - 1)}.$$

If we fix n , d and k and increase r , then the storage efficiency increases.

VII. TWO FAMILIES OF EXPLICIT COOPERATIVE REGENERATING CODES

In this section we present two families of explicit constructions of optimal cooperative regenerating codes for exact repair, one for MSCR and one for MBCR. The constructed regenerated codes are systematic, meaning that the native data packets are stored somewhere in the storage network. Hence, if a data collector is interested in part of the data file, he/she can contact some particular storage nodes and download directly without any decoding. Both constructions are for the case $d = k$. We note that all single-failure regenerating codes for $d = k$ are trivial, but in the multi-failure case, something interesting can be done when $d = k$.

A. Construction of MSCR Codes for Exact Repair

We divide the data file into chunks. Each chunk contains B elements in a finite field \mathbb{F}_q . The size of the finite field will be determined later. Each node stores r elements in \mathbb{F}_q . The parameters of the cooperative regenerating code are

$$\begin{aligned} d &= k, \quad B = kr, \quad n \geq d + r, \\ \alpha &= r, \quad \gamma = d + r - 1. \end{aligned}$$

It matches the MSCR point.

We need an (n, k) MDS code as a building block. The code length is equal to the number of storage nodes. This can be furnished by Reed-Solomon code for instance, as long as the finite field size q is larger than the total number of storage nodes n [33]. We let \mathbf{G} be the $n \times k$ generating matrix of an (n, k) MDS code over \mathbb{F}_q . For $i = 1, 2, \dots, n$, let the i -th row of \mathbf{G} be denoted by \mathbf{g}_i . By the defining property of MDS code, every set of k columns of \mathbf{G} form a nonsingular matrix. We group the kr finite field elements in a chunk of data into r k -dimensional column vectors, $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_r$. The data stored in the i -th storage node are $\mathbf{g}_i \mathbf{m}_j$, for $j = 1, 2, \dots, r$. We will treat a finite field element as a packet.

Suppose that a data collector connects to storage node i_1, i_2, \dots, i_k . It downloads kr packets $\mathbf{g}_{i_\ell} \mathbf{m}_j$, for $j = 1, 2, \dots, r$ and $\ell = 1, 2, \dots, k$. If we put the k symbols $\mathbf{g}_{i_\ell} \mathbf{m}_j$, $\ell = 1, 2, \dots, k$, together as a column vector, then this vector can be expressed as

$$\begin{bmatrix} \mathbf{g}_{i_1} \\ \mathbf{g}_{i_2} \\ \vdots \\ \mathbf{g}_{i_k} \end{bmatrix} \cdot \mathbf{m}_j.$$

The $k \times k$ matrix in the line above is non-singular by the MDS property. We can thus solve for \mathbf{m}_j . This establishes the (n, k) recovery property.

Suppose that nodes i_1, i_2, \dots, i_r fail. We want to repair them exactly with repair-bandwidth $d+r-1$ per newcomer. In the first phase, the r newcomers have to agree upon an ordering among themselves. For $\ell = 1, 2, \dots, r$, the ℓ -th newcomer connects to from any d surviving storage nodes, say nodes $\nu_{\ell,1}, \nu_{\ell,2}, \dots, \nu_{\ell,d}$, and download $\mathbf{g}_{\nu_{\ell,x}} \mathbf{m}_\ell$ from node $\nu_{\ell,x}$, for $x = 1, 2, \dots, d$. Then newcomer ℓ can decode \mathbf{m}_ℓ by the MDS property by the end of the first phase. The total packets generated is rd .

In the second phase, newcomer i_y computes and sends $\mathbf{g}_{i_x} \mathbf{m}_{i_y}$ to newcomer i_x , for $x \neq y$. Newcomer i_x stores $\mathbf{g}_{i_x} \mathbf{m}_{i_x}$ at the end. A total of $r(r-1)$ packets are required in the second phase. The r newcomers are regenerated with repair-bandwidth per newcomer equal to $d+r-1$.

B. Construction of MBCR Codes for Exact Repair

The parameters of the cooperative regenerating code are

$$\begin{aligned} d &= k, \quad B = k(k+r), \\ n &= d+r, \quad \alpha = \gamma = 2d+r-1. \end{aligned}$$

It matches the MBCR point.

Each chunk of data consists of $B = k(2d+r-k) = kn$ data packets, considered as elements in $GF(q)$. In each chunk

let the kn data packets be $x_0, x_1, \dots, x_{kn-1}$. We divide them into n groups. The first group consists of x_0, x_1, \dots, x_{k-1} , the second group consists of $x_k, x_{k+1}, \dots, x_{2k-1}$, and so on. For notational convenience, we let the column vector $\mathbf{x}_j = [x_{(j-1)k}, x_{(j-1)k+1}, \dots, x_{(j-1)k+k-1}]^T$ represent the data packets in the j -th group ($1 \leq j \leq n$). (We use superscript T to denote the transpose operator.)

For $i = 1, 2, \dots, n$, we construct the content of node i as follows. We first put the k data packets in the i -th group \mathbf{x}_i into node i and then $n-1$ parity-check packets

$$\mathbf{v}_1 \cdot \mathbf{x}_{i \oplus 1}, \mathbf{v}_2 \cdot \mathbf{x}_{i \oplus 2}, \dots, \mathbf{v}_{n-1} \cdot \mathbf{x}_{i \oplus (n-1)}$$

into node i , where “ \cdot ” is the dot product of vectors and \oplus is modulo- n addition defined by

$$x \oplus y := \begin{cases} x + y & \text{if } x + y \leq n, \\ x + y - n & \text{if } x + y > n. \end{cases}$$

Here \mathbf{v}_j ($j = 1, 2, \dots, n-1$) are row vectors in a $(n-1) \times k$ generating matrix, $\mathbf{G} = [\mathbf{v}_1^T \mathbf{v}_2^T \dots \mathbf{v}_{n-1}^T]^T$, of an MDS code over $GF(q)$ of length $n-1$ and dimension k . By the defining property of MDS code, any k columns of \mathbf{G} are linearly independent of $GF(q)$.

As for the file reconstruction processing, suppose without loss of generality that a data collector connects to nodes $1, 2, \dots, k$. The systematic packets x_0, x_1, \dots, x_{k-1} in the first k groups can be downloaded directly, because they are stored in node 1 to node k uncoded. The j -th group of data packets ($j > k$) (the components in vector \mathbf{x}_j) can be reconstructed from $\mathbf{v}_{j-1} \cdot \mathbf{x}_j, \mathbf{v}_{j-2} \cdot \mathbf{x}_j, \dots, \mathbf{v}_{j-k} \cdot \mathbf{x}_j$, by the MDS property. A data collector connecting to any other k storage nodes can decode similarly.

As for the cooperative repair processing, suppose without loss of generality that nodes $k+1$ to n fail at the same time. The repair process proceeds as follows.

- Step 1: For $i = 1, 2, \dots, k$, node i computes $\mathbf{v}_{n+i-j} \cdot \mathbf{x}_i$ and sends it to newcomer j , for $j = k+1, k+2, \dots, n$.
- Step 2: For $j = k+1, k+2, \dots, n$, newcomer j downloads k packets $\mathbf{v}_{j-1} \cdot \mathbf{x}_j, \mathbf{v}_{j-2} \cdot \mathbf{x}_j, \dots, \mathbf{v}_{j-k} \cdot \mathbf{x}_j$ from nodes 1 to k .
- Step 3: For $j = k+1, k+2, \dots, n$, newcomer j can solve for the systematic packets in \mathbf{x}_j . Then node j sends $\mathbf{v}_{j+i} \cdot \mathbf{x}_j$ to node $n-i+1$, for $i = 1, 2, \dots, n-j$, and sends $\mathbf{v}_i \cdot \mathbf{x}_j$ to node $j-i$, for $i = 1, 2, \dots, j-k-1$.

In steps 1 and 2, a total of $2k(n-k) = 2kr$ packets are transmitted. In step 3, each newcomer transmits $r-1$ packets. The total number of packets required in the whole repair process is $2kr+r(r-1) = r(2d+r-1)$. The repair-bandwidth per newcomer is therefore $2d+r-1$ packets.

We illustrate this MBCR construction by an example. The parameters are $n = 5, k = d = 3$ and $r = 2$. A file is divided into $B = 15$ packets, each containing equal number of bits. Let the packets be x_1, x_2, \dots, x_{15} . Each node stores $\alpha = 7$

Node					
1	x_1, x_2, x_3	x_4	x_8	x_{12}	$x_{13} + x_{14} + x_{15}$
2	$x_1 + x_2 + x_3$	x_4, x_5, x_6	x_7	x_{11}	x_{15}
3	x_3	$x_4 + x_5 + x_6$	x_7, x_8, x_9	x_{10}	x_{14}
4	x_2	x_6	$x_7 + x_8 + x_9$	x_{10}, x_{11}, x_{12}	x_{13}
5	x_1	x_5	x_9	$x_{10} + x_{11} + x_{12}$	x_{13}, x_{14}, x_{15}

TABLE I

AN MBCR CODE FOR $d = 5$, $k = 3$ AND $r = 2$. THE PACKETS IN EACH ROW ARE THE CONTENT OF THE CORRESPONDING NODE.

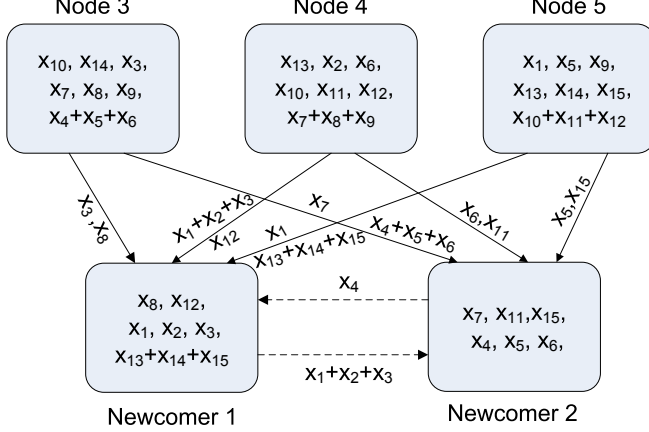


Fig. 13. Cooperative recovery of nodes 1 and 2.

1 and sends x_5 and x_{15} to newcomer 2. The contents of the two newcomers after the first phase of recovery are shown in Fig. 13. Notice that newcomer 1 is able to compute x_3 by adding x_1 and x_2 to the parity-check packet $x_1 + x_2 + x_3$. Similarly, newcomer 2 computes x_4 from the three packets x_5 , x_6 and $x_4 + x_5 + x_6$.

In the second phase, newcomer 1 sends $x_1 + x_2 + x_3$ to newcomer 2, and in return, newcomer 2 sends x_4 to newcomer 1. The data exchange in the second phase is indicated by the dashed arrows in Fig. 13. This completes the joint repair of node 1 and 2. The number of packet transmissions per newcomer is equal to $\gamma = 7$.

Any double failures can be recovered similarly with repair-bandwidth per newcomer equal to 7. This is indeed optimal by Corollary 18.

packets. We use the following matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

over \mathbb{F}_2 as the generating matrix of the MDS code. The content of the storage nodes is shown in Table. I:

Each node contains six uncoded packets, and one parity-check packet. The addition in the above table is component-wise binary addition. In this example the parity-check packets are computed simply by exclusive-OR (XOR). For example in node 1, we XOR packets x_{13} , x_{14} and x_{15} to obtain the parity-check packet. We note that there is a cyclic symmetry in the encoding. The indices of the packets in node 2 can be obtained by adding 3 to the corresponding indices in node 1 and reduce modulo 15. From node 2, we can add 3 modulo 15 to the packet indices and get the packets in node 3, and so on.

It can be verified that it satisfies the $(5, 3)$ recovery property. We note that this code is *not* MDS, as each node stores more than $15/3 = 5$ packets.

Suppose that node 1 and node 2 fail. Newcomers 1 and 2 regenerate the failed node by downloading some packets from the other $d = 3$ surviving storage nodes. The packets sent from the surviving nodes to the two newcomers are shown in Fig. 13. Each surviving node transmits two packets to each of the two newcomers. Node 3 sends x_3 and x_8 to newcomer 1, and sends x_7 and $x_4 + x_5 + x_6$ to newcomer 2. Node 4 sends $x_1 + x_2 + x_3$ and x_{12} to newcomer 1 and sends x_6 and x_{11} to newcomer 2. Node 5 sends x_1 and $x_{13} + x_{14} + x_{15}$ to newcomer

VIII. CONCLUDING REMARKS

One of the technical difficulties in this paper is to deal with an unbounded information flow graph. The number of sink nodes may be infinite. By exploiting the structure of the information flow graph, we show that the points on the fundamental tradeoff curve can be attained by linear cooperative regenerating codes. Furthermore, we can work over a fixed finite field for arbitrarily large number of repairs.

The proof technique invokes a concept called “submodular flow” from combinatorial optimization, which can be regarded as a generalization of the max-flow-min-cut theorem. This mathematical tool has also been used in [34] to determine the capacity of linear deterministic model of relay networks. Indeed, the information flow graph studied in this paper share some common features with the linear deterministic model for wireless relay network as proposed in [35]. A closely related notion, called “linking system”, finds application in perfect secret key agreement [36], [37], and in wireless information flow [38], [39].

Recently, constructions of cooperative regenerating codes achieving the MBCR point are given in [40] and [41]. Further explicit construction for other parameters is an interesting research problem for future studies.

ACKNOWLEDGEMENTS

We would like to thank Chung Chan and Frédérique Oggier for useful discussions, and the anonymous reviewers for their careful readings.

APPENDIX A

THE MSCR POINT UNDER NON-HOMOGENEOUS TRAFFIC

Homogeneous traffic is assumed in the main text of this paper; a newcomer downloads equal amount of data from the d surviving nodes, and each pair of newcomers exchange equal amount of data. We show in this appendix that the assumption of homogeneous traffic is not essential at the minimum-storage point, i.e., the repair-bandwidth cannot be decreased when $\alpha = B/k$ if property of homogeneous traffic is relaxed.

Suppose that each newcomer connects to d surviving nodes, and the total number of packets transmitted in the first phase be $rd\bar{\beta}_1$. The average number of packets per link is thus $\bar{\beta}_1$. Likewise, we let the total number of packets in the second phase be $r(r-1)\bar{\beta}_2$, so that the average number of packets per link is $\bar{\beta}_2$. We call this the non-homogeneous traffic model. Naturally, it contains the homogeneous traffic model as a special case if each newcomer download $\bar{\beta}_1$ packets per each link in the first phase, and $\bar{\beta}_2$ packets per each link in the second phase. In the non-homogeneous model, it is only required the traffic in the first (resp. second) phases of all repair processes are identical.

Theorem 19. *Under the non-homogeneous traffic model, the average repair-bandwidth per newcomer is lower bounded by $B(d+r-1)/(k(d+r-k))$ at the minimum-storage point, i.e., $B/k = \alpha$.*

Proof: Consider the scenario where nodes 1 to r fail in stage 1. Suppose that each newcomer connects to nodes $r+1$, $r+2, \dots, r+d$. We will derive a lower bound on the repair-bandwidth.

For $i = r+1, r+2, \dots, r+d$ and $j = 1, 2, \dots, r$, let the capacity of the link from surviving node i to newcomer j be $\beta_1(i, j)$. Let the capacity of the link from In_{j_1} to Mid_{j_2} , for $j_1 \neq j_2$ is $\beta_2(j_1, j_2)$. The average link capacities in the first and second phase can be written respectively as

$$\bar{\beta}_1 = \frac{1}{dr} \sum_{i=r+1}^{r+d} \sum_{j=1}^r \beta_1(i, j)$$

$$\bar{\beta}_2 = \frac{1}{r(r-1)} \sum_{j_2=1}^r \sum_{\substack{j_1=1 \\ j_1 \neq j_2}}^r \beta_2(j_1, j_2).$$

The repair-bandwidth per newcomer is thus

$$\frac{1}{r} \sum_{i=r+1}^{r+d} \sum_{j=1}^r \beta_1(i, j) + \frac{1}{r} \sum_{j_2=1}^r \sum_{\substack{j_1=1 \\ j_1 \neq j_2}}^r \beta_2(j_1, j_2) = d\bar{\beta}_1 + (r-1)\bar{\beta}_2.$$

We consider a set of data collectors. Each data collector in this set connects to one of the r new nodes which are regenerated in stage 1, and $k-1$ nodes among nodes $r+1$ to $r+d$. Let DC be such a data collector, connecting to node j , for some $j \in \{1, 2, \dots, r\}$, and node $i_1, i_2, \dots, i_{k-1} \in \{r+1, r+2, \dots, r+d\}$. Consider the cut $(\mathcal{W}, \bar{\mathcal{W}})$, with

$$\bar{\mathcal{W}} = \{\text{DC}, \text{In}_j, \text{Mid}_j, \text{Out}_j, \text{Out}_{i_1}, \text{Out}_{i_2}, \dots, \text{Out}_{i_{k-1}}\}.$$

An example is given in Fig. 14, with $\bar{\mathcal{W}}$ drawn in shaded color.

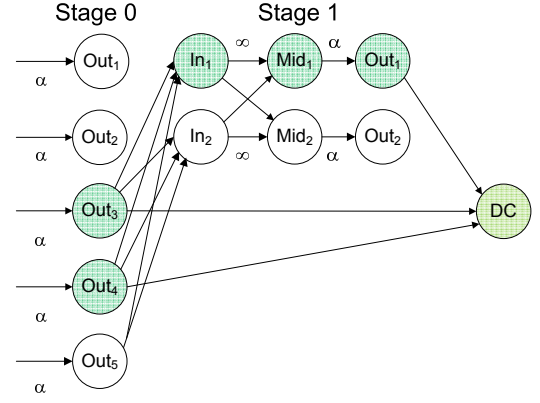


Fig. 14. A DC connects to one storage node among the first r nodes and $k-1$ storage nodes among the nodes $r+1$ to $r+d$.

There are $r \binom{d}{k-1}$ distinct choices. Each choice yields a cut-set upper bound on the file size B . If we sum over all $r \binom{d}{k-1}$ such inequalities, we obtain

$$r \binom{d}{k-1} B \leq r \binom{d}{k-1} (k-1)\alpha$$

$$+ \binom{d-1}{k-1} \sum_{i=r+1}^{r+d} \sum_{j=1}^r \beta_1(i, j)$$

$$+ \binom{d}{k-1} \sum_{j_2=1}^r \sum_{\substack{j_1=1 \\ j_1 \neq j_2}}^r \beta_2(j_1, j_2).$$

The first term on the right-hand side of the inequality comes from the fact that each of the $r \binom{d}{k-1}$ inequalities contributes $(k-1)\alpha$. For the second term, we note that the link from node i to node j , (for $r+1 \leq i \leq r+d$, $1 \leq j \leq r$) contributes $\beta_1(i, j)$ to the summation if node $i \in \mathcal{W}$ and node $j \in \bar{\mathcal{W}}$. There are $\binom{d-1}{k-1}$ choices for the “out” nodes to be included in $\bar{\mathcal{W}}$. Hence for each i, j , the term $\beta_1(i, j)$ is multiplied by $\binom{d-1}{k-1}$. By similar argument we can obtain the third term.

After dividing both sides by $r \binom{d}{k-1}$, we obtain

$$B \leq (k-1)\alpha + (d-k+1)\bar{\beta}_1 + (r-1)\bar{\beta}_2. \quad (42)$$

In the rest of the proof we distinguish two cases.

Case 1: $k \geq r$. Consider the class of data collectors who download from nodes 1 to r , and $k-r$ nodes among nodes $r+1$ to $r+d$. For a data collector DC, say connecting to nodes 1 to r , and $i_1, i_2, \dots, i_{k-r} \in \{r+1, r+2, \dots, r+d\}$, we compute the capacity of the cut $(\mathcal{W}, \bar{\mathcal{W}})$ with $\bar{\mathcal{W}}$ specified by

$$\bar{\mathcal{W}} = \{\text{DC}, \text{Out}_{i_1}, \text{Out}_{i_2}, \dots, \text{Out}_{i_{k-r}}\} \cup \bigcup_{j=1}^r \{\text{In}_j, \text{Mid}_j, \text{Out}_j\}.$$

If we sum over the $\binom{d}{k-r}$ inequalities arising from these cuts, we get

$$\binom{d}{k-r} B \leq \binom{d}{k-r} (k-r)\alpha$$

$$+ \binom{d-1}{k-r} \sum_{i=r+1}^{r+d} \sum_{j=1}^r \beta_1(i, j).$$

Upon dividing both sides by $\binom{d}{k-r}$, we obtain

$$B \leq (k-r)\alpha + (d-k+r)r\bar{\beta}_1. \quad (43)$$

With $\alpha = B/k$, we infer from (42) and (43) that

$$d\bar{\beta}_1 + (r-1)\bar{\beta}_2 \geq \frac{B(d+r-1)}{k(d+r-k)}. \quad (44)$$

Case 2: $k < r$. Consider the class of data collectors who connects to k nodes among nodes 1 to r . To a data collector DC connecting to $j_1, j_2, \dots, j_k \in \{1, 2, \dots, r\}$, we associate it with the cut $(\mathcal{W}, \bar{\mathcal{W}})$ with $\bar{\mathcal{W}}$ given by

$$\bar{\mathcal{W}} = \{\text{DC}\} \cup \bigcup_{\ell=1}^k \{\text{In}_{i_\ell}, \text{Mid}_{i_\ell}, \text{Out}_{i_\ell}\}.$$

The sum of the $\binom{r}{k}$ resulting upper bounds on B is

$$\begin{aligned} \binom{r}{k} B &\leq \binom{r-1}{k-1} \sum_{i=1}^r \sum_{j=r+1}^{r+d} \beta_1(i, j) \\ &\quad + \binom{r-1}{k-1} \sum_{j_2=1}^r \sum_{\substack{j_1=1 \\ j_1 \neq j_2}}^r \beta_2(j_1, j_2). \end{aligned}$$

After dividing both sides by $\binom{r}{k}$, we get

$$B \leq kd\bar{\beta}_1 + k(r-k)\bar{\beta}_2. \quad (45)$$

From (42) and (45), we can deduce (44) as well. We conclude that in the non-homogeneous traffic case, the repair-bandwidth per newcomer cannot be smaller than $B(d+r-1)/(k(d+r-k))$ when $\alpha = B/k$. ■

APPENDIX B PROOF OF LEMMA 10

We first show that it is sufficient to verify that the condition (39) in Lemma 10 holds for subsets \mathcal{S} in the form of

$$\mathcal{S} = \left(\bigcup_{i \in \mathcal{A}} \{\text{In}_i, \text{Mid}_i, \text{Out}_i\} \right) \cup \left(\bigcup_{j \in \mathcal{B}} \{v_j, \text{Out}_j\} \right), \quad (46)$$

where \mathcal{A} is a subset of $\{1, 2, \dots, r\}$ and \mathcal{B} is a subset of $\{r+1, r+2, \dots, n\}$.

An example of a subset \mathcal{S} in the form of (46) is illustrated in Fig. 11.

(a) Suppose for some $j \in \{r+1, r+2, \dots, n\}$, \mathcal{S} contains v_j but does not contain Out_j . Then the directed edge $e = (v_j, \text{Out}_j)$ is in $\Delta_{\mathcal{S}}^{\text{out}}$, and make a contribution of h_j to the term $\text{lb}(\Delta_{\mathcal{S}}^{\text{out}})$. But the inequality $\text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) \leq \rho(\mathcal{S})$ holds if and only if

$$\begin{aligned} \text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) - h_j &\leq \rho(\mathcal{S}) - h_j \\ \Leftrightarrow \text{lb}(\Delta_{\mathcal{S} \cup \{\text{Out}_j\}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S} \cup \{\text{Out}_j\}}^{\text{in}}) &\leq \rho(\mathcal{S} \cup \{\text{Out}_j\}) \end{aligned}$$

holds.

An analogous argument shows that if \mathcal{S} contains Out_j but does not contains v_j for some $j \in \{r+1, r+2, \dots, n\}$, then the validity of $\text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) \leq \rho(\mathcal{S})$ is equivalent to

$$\begin{aligned} \text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) + h_j &\leq \rho(\mathcal{S}) + h_j \\ \Leftrightarrow \text{lb}(\Delta_{\mathcal{S} \setminus \{\text{Out}_j\}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S} \setminus \{\text{Out}_j\}}^{\text{in}}) &\leq \rho(\mathcal{S} \setminus \{\text{Out}_j\}). \end{aligned}$$

Hence it is sufficient to consider subset \mathcal{S} which either contains both v_j and Out_j , or none of them.

(b) Suppose that \mathcal{S} contains Mid_i for some $i \in \{1, 2, \dots, r\}$. Since the link from In_i to Mid_i has infinite upper bound, the left-hand side of (39) is equal to $-\infty$ if In_i is not included in \mathcal{S} . Then the inequality in (39) holds trivially. We can assume without generality that $\text{In}_i \in \mathcal{S}$ if $\text{Mid}_i \in \mathcal{S}$.

On the other hand, if Out_i is not included in \mathcal{S} , the inequality $\text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) \leq \rho(\mathcal{S})$ is implied by

$$\text{lb}(\Delta_{\mathcal{S} \cup \{\text{Out}_i\}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S} \cup \{\text{Out}_i\}}^{\text{in}}) \leq \rho(\mathcal{S} \cup \{\text{Out}_i\}),$$

because

$$\begin{aligned} \text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) &= \text{lb}(\Delta_{\mathcal{S} \cup \{\text{Out}_i\}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S} \cup \{\text{Out}_i\}}^{\text{in}}) \\ &\leq \rho(\mathcal{S} \cup \{\text{Out}_i\}) \\ &= \rho(\mathcal{S}) - h_i \leq \rho(\mathcal{S}). \end{aligned}$$

We can thus assume that if Mid_i is in \mathcal{S} then Out_i are also in \mathcal{S} .

(c) Suppose that for some $i \in \{1, 2, \dots, r\}$ \mathcal{S} contains In_i but does not contain Mid_i and Out_i . In this case the inequality $\text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) \leq \rho(\mathcal{S})$ is implied by

$$\begin{aligned} \text{lb}(\Delta_{\mathcal{S} \cup \{\text{Mid}_i, \text{Out}_i\}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S} \cup \{\text{Mid}_i, \text{Out}_i\}}^{\text{in}}) \\ \leq \rho(\mathcal{S} \cup \{\text{Mid}_i, \text{Out}_i\}), \end{aligned}$$

because

$$\begin{aligned} \text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) &= \text{lb}(\Delta_{\mathcal{S} \cup \{\text{Mid}_i, \text{Out}_i\}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S} \cup \{\text{Mid}_i, \text{Out}_i\}}^{\text{in}}) \\ &\leq \rho(\mathcal{S} \cup \{\text{Mid}_i, \text{Out}_i\}) \\ &= \rho(\mathcal{S}) - h_j \leq \rho(\mathcal{S}). \end{aligned}$$

(d) Suppose that \mathcal{S} contains Out_i but does not contain Mid_i for some $i \in \{1, 2, \dots, r\}$. In this case, the inequality $\text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) \leq \rho(\mathcal{S})$ is implied by the following two inequalities

$$\text{lb}(\Delta_{\mathcal{S} \setminus \{\text{Out}_i\}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S} \setminus \{\text{Out}_i\}}^{\text{in}}) \leq \rho(\mathcal{S} \setminus \{\text{Out}_i\}), \quad (47)$$

$$\text{lb}(\Delta_{\{\text{Out}_i\}}^{\text{out}}) - \text{ub}(\Delta_{\{\text{Out}_i\}}^{\text{in}}) \leq \rho(\{\text{Out}_i\}). \quad (48)$$

(We can add (47) and (48) to obtain the inequality in (39).) The inequality in (48) is simply equivalent to $-\alpha \leq -h_i$, which holds by the assumption on \mathbf{h} . Hence, the checking of (39) amounts to the check of (47).

Paragraphs (b), (c) and (d) above imply that we can consider \mathcal{S} which contains either none of In_i , Mid_i and Out_i , or all of them. This completes the proof of the claim.

Now, we prove that the inequality in (39) is valid for subset \mathcal{S} in the form of (46). We let the cardinality of \mathcal{A} and \mathcal{B} be a and b respectively. Obviously we have $a \leq r$ and $b \leq n-r$.

In the following we will use $(x)^+$ as a short-hand notation for $\max(0, x)$.

Because $\text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) = 0$, we have

$$\begin{aligned} \text{lb}(\Delta_{\mathcal{S}}^{\text{out}}) - \text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) &= -\text{ub}(\Delta_{\mathcal{S}}^{\text{in}}) \\ &\leq -a((d-b)^+ \beta_2 + (r-a)\beta_1) \\ &= -a(2(d-b)^+ + (r-a)). \end{aligned}$$

It suffices to show that

$$-a(2(d-b)^+ + (r-a)) \leq \theta_b - \mathbf{h}(\mathcal{A}) - \mathbf{h}(\mathcal{B}) = \sigma(\mathcal{S}),$$

As $\mathbf{h}(\mathcal{A}) + \mathbf{h}(\mathcal{B})$ is less than or equal to θ_{a+b} by hypothesis, it is sufficient to show that

$$-a(2(d-b)^+ + r - a) \leq \theta_b - \theta_{a+b}$$

or equivalently

$$\theta_{a+b} - \theta_b \leq a(2(d-b)^+ + r - a) \quad (49)$$

We prove the asserted inequality in (49) by distinguishing three cases.

Case 1, $a + b \leq k$: We first note that for $j \leq k$, we have

$$\begin{aligned} \theta_j - \theta_{j-1} &= \begin{cases} \alpha & \text{if } 0 < j \leq z \\ \alpha - 2(j - z - 1) & \text{if } z < j \leq k \end{cases} \\ &\leq \alpha - 2(j - z - 1) \quad \text{for } 0 < j \leq k. \end{aligned}$$

Hence, for $0 < j \leq k$, we have the following upper bound

$$\theta_j - \theta_{j-1} \leq 2(d - z) + r - 1 - 2(j - z - 1) = 2(d - j) + r + 1.$$

Summing the above inequality for j from $b + 1$ to $a + b$, we obtain

$$\begin{aligned} \theta_{a+b} - \theta_b &= \sum_{j=b+1}^{a+b} \theta_j - \theta_{j-1} \\ &\leq \sum_{j=b+1}^{a+b} [2(d - j) + r + 1] \\ &= a(2(d - b) + r - a) \\ &= a(2(d - b)^+ + r - a). \end{aligned}$$

We have use the assumptions that $d \geq k$ and $k \geq b$ to derive the last equality.

Case 2, $b \leq k \leq a + b$ Since $\theta_k = \theta_{k+1} = \dots = \theta_{a+b}$ in this case, we have

$$\begin{aligned} \theta_{a+b} - \theta_b &= \theta_k - \theta_b \\ &\leq (k - b)(2(d - b) + r - (k - b)) \\ &\leq a(2(d - b) + r - a) \\ &= a(2(d - b)^+ + r - a). \end{aligned}$$

The first inequality is obtained from the previous case.

Case 3, $k \leq b$: (49) holds because $\theta_{a+b} - \theta_b = 0$ on the left-hand side, while the right-hand side is non-negative.

This completes the verification that condition (39) in Lemma 10 holds.

APPENDIX C PROOF OF THEOREM 12

Proof: (sketch) Let \mathcal{Q} be the set of vectors in \mathbb{Z}_+^n which are majorized by the vector in (40). We can check that the vectors in \mathcal{Q} belong to the polymatroid associated with rank function $f(\mathcal{S}) = \varphi_{|\mathcal{S}|}$, where φ_x is defined by

$$\varphi_x := \min(k, x)\alpha - \sum_{i=0}^{\min(k, x) - k + \ell r} \lceil i/r \rceil r \quad (50)$$

for $x = 0, 1, 2, \dots, n$. (The summation is defined as 0 if the upper limit is negative.) The sum of the components in (40) is equal to B ,

$$\begin{aligned} (k - \ell r)\alpha + r \sum_{i=1}^{\ell} (\alpha - ir) &= k\alpha - r^2 \sum_{i=1}^{\ell} i \\ &= k\alpha - r^2 \frac{\ell(\ell + 1)}{2} \\ &= B = \theta_k, \end{aligned}$$

and the difference between consecutive ϕ_x is equal to the x -th components in (40),

$$\varphi_x - \varphi_{x-1} = \begin{cases} \alpha & \text{if } 0 < x \leq k - \ell r \\ \alpha - r & \text{if } k - \ell r < x \leq k - (\ell - 1)r \\ \alpha - 2r & \text{if } k - (\ell - 1)r < x \leq k - (\ell - 2)r \\ \vdots & \vdots \\ \alpha - \ell r & \text{if } k - r < x \leq k \\ 0 & \text{if } k < x \leq n \end{cases}$$

for $x = 1, 2, \dots, n$.

The proof is along the same line as in the proof of Theorem 9. We define the same submodular function on the vertex set \mathcal{V}' of this auxiliary graph, except that β_1 is equal to 1 and α is equal to $d + r(\ell + 1) - k$ in this proof. We want to show that the inequality $\text{lb}(\Delta_S^{\text{out}}) - \text{ub}(\Delta_S^{\text{in}}) \leq \sigma(\mathcal{S})$ holds for all subsets \mathcal{S} which is in the form of (46).

Analogous to (49), we need to show that

$$\varphi_{a+b} - \varphi_b \leq a((d - b)^+ + r - a). \quad (51)$$

for $0 \leq a \leq r$ and $0 \leq b \leq d$.

As in the proof of Theorem 9, we consider three cases. (1) $a + b \leq k$, (2) $b \leq k < a + b$ and (3) $k < b$.

Case (1) $a + b \leq k$. Note that the difference $\varphi_{a+b} - \varphi_b$, i.e., the x -th component in (51) can be written as

$$\alpha - \left\lceil \frac{x - k + \ell r}{r} \right\rceil r \quad (52)$$

We need to take the sum of (52) for $b < x \leq b + a$. Note that the value of (52) is constant for r consecutive values of x . Since a is no larger than r , $\lceil (x - k + \ell r)/r \rceil$ assumes at most two values for $b < x \leq b + a$. We further divide into two cases.

First subcase: $\lceil (x - k + \ell r)/r \rceil$ is constant for $b < x \leq b + a$. For x in this range, $\lceil (x - k + \ell r)/r \rceil$ is equal to $\lceil (a + b - k + \ell r)/r \rceil$. We have

$$\begin{aligned} &a((d - b)^+ + r - a) - (\varphi_{a+b} - \varphi_b) \\ &= a((d - b)^+ + r - a) - \sum_{x=b+1}^{a+b} \left(\alpha - \left\lceil \frac{a + b - k + \ell r}{r} \right\rceil r \right) \\ &\geq a(d - b + r - a - \alpha + (a + b - k + \ell r)) \\ &= a(d + (\ell + 1)r - k - \alpha) = 0. \end{aligned}$$

Second subcase: $\lceil (x - k + \ell r)/r \rceil$ is not constant for $b < x \leq b + a$. Suppose that $a + b > k - \ell r + \xi r$ and $b \leq k - \ell r + \xi r$

for some integer ξ . We have

$$\left\lceil \frac{x - k + \ell r}{r} \right\rceil = \begin{cases} \xi & \text{for } b < x \leq k - \ell r + \xi r \\ \xi + 1 & \text{for } k - \ell r + \xi r < x \leq a + b. \end{cases}$$

For the ease of presentation, we use δ to stand for $a + b - (k - \ell r + \xi r)$, and let Y be $d - b + r - a$. δ is positive construction. With these notations, we get

$$\begin{aligned} & a((d - b)^+ + r - a) - (\varphi_{a+b} - \varphi_b) \\ & \geq aY - (\varphi_{a+b} - \varphi_b) \\ & = aY - (a - \delta)(\alpha - \xi r) - \delta(\alpha - (\xi + 1)r) \\ & = (a - \delta)(Y - \alpha + \xi r) + \delta(Y - \alpha + (\xi + 1)r) \end{aligned}$$

Since

$$Y - \alpha + \xi r = d - b + r - a - d - r\ell - r + k + \xi r = -\delta,$$

we get

$$\begin{aligned} & a((d - b)^+ + r - a) - (\varphi_{a+b} - \varphi_b) \\ & \geq -(a - \delta)\delta + \delta(r - \delta) \\ & = \delta(r - a) \geq 0. \end{aligned}$$

This proves case (1). The proofs of case (2) and case (3) are similar to those in Theorem 9 and are omitted.

The proof proceeds by applying Theorem 3 to recursively construct flow on the modified information flow graph. ■

APPENDIX D PROOF OF THEOREM 15

We first prove two theorems which will be useful later. The first one is about the envelope of straight lines.

Lemma 20. *Let $y = m_i x + b_i$ for $i = 1, 2, \dots, N$, be N straight lines in the x - y plane, satisfying the following conditions:*

(a) *The slopes are negative with decreasing magnitudes, i.e.,*

$$-m_1 > -m_2 > \dots > -m_N > 0.$$

(b) *For $j = 1, 2, \dots, N - 1$, the x -coordinates of the intersection point of $y = m_j x + b_j$ and $y = m_{j+1} x + b_{j+1}$, denoted by x_j , are strictly increasing, i.e.,*

$$x_1 < x_2 < \dots < x_{N-1}.$$

Then we have

$$\max_{j=1, \dots, N} (m_j x + b_j) = \begin{cases} m_1 x + b_1 & \text{for } 0 \leq x < x_1 \\ m_j x + b_j & \text{for } x_{j-1} \leq x < x_j \\ & j = 2, 3, \dots, N - 1 \\ m_N x + b_N & \text{for } x > x_{N-1}. \end{cases}$$

We notice that x_j is equal to $(b_j - b_{j+1}) / (m_{j+1} - m_j)$, and is positive by the assumptions in (a).

Proof: Since the slope of L_i is more negative than the slope of L_{i+1} , for $i = 1, 2, \dots, N - 1$, we see that

$$\begin{cases} m_i x + b_i > m_{i+1} x + b_{i+1} & \text{for } x < x_i \\ m_i x + b_i = m_{i+1} x + b_{i+1} & \text{for } x = x_i \\ m_i x + b_i < m_{i+1} x + b_{i+1} & \text{for } x > x_i. \end{cases}$$

For x between 0 and x_1 , we have $x < x_i$ for all i . Hence $m_1 x + b_1 > m_j x + b_j$ for $j = 2, 3, \dots, N - 1$. Therefore,

$$\max_{i=1, \dots, N} (m_i x + b_i) = m_1 x + b_1$$

for $x < x_1$.

Consider x in the interval $[x_{i-1}, x_i]$, for some $i = 2, 3, \dots, N - 1$. Since $x \geq x_{i-1} > x_{i-2} > \dots > x_1$, we get

$$m_i x + b_i > m_{i-1} x + b_{i-1} > \dots > m_1 x + b_1.$$

On the other hand, since $x < x_i < \dots < x_{N-1}$, we get

$$m_i x + b_i > m_{i+1} x + b_{i+1} > \dots > m_N x + b_N.$$

Therefore

$$\max_{j=1, \dots, N} (m_j x + b_j) = m_i x + b_i$$

for $x_{i-1} \leq x < x_i$.

The proof of the last case $x > x_{N-1}$ is similar and omitted. ■

The second lemma is a special case of duality and gives a sufficient condition for optimality in linear program.

Lemma 21. *Consider the linear programming with objective function $c_1 x + c_2 y$, where x and y are variables and c_1 and c_2 are constants, subject to constraints $a_{i1} x + a_{i2} y \geq b_i$, for $i = 1, 2, \dots, N$, and $x, y \geq 0$. If (\bar{x}, \bar{y}) is a point which*

(a) *satisfies all constraints,*

(b) *attains equality in two particular constraints whose slopes are distinct, and the slope of the objective function is between these two slopes.*

then (\bar{x}, \bar{y}) is the optimal solution to the linear programming problem.

Proof: It suffices to show that the value of the objective function cannot be smaller than $c_1 \bar{x} + c_2 \bar{y}$ without violating any constraints. By re-indexing the constraints, suppose without loss of generality that (\bar{x}, \bar{y}) satisfies equations $a_{i1} x + a_{i2} y = b_i$ for $i = 1, 2$, and

$$a_{11}/a_{12} > a_{21}/a_{22}, \quad (53)$$

$$a_{11}/a_{12} \geq c_1/c_2 \geq a_{21}/a_{22}. \quad (54)$$

Let \mathbf{A} be the matrix

$$\mathbf{A} := \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

The matrix \mathbf{A} is invertible by (53), and we have

$$\mathbf{A} \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Let (p, q) be the solution to

$$\begin{bmatrix} p & q \end{bmatrix} \mathbf{A} = \begin{bmatrix} c_1 & c_2 \end{bmatrix}.$$

We can solve for p and q and see that the solutions

$$p = \frac{c_1 a_{22} - c_2 a_{21}}{a_{11} a_{22} - a_{12} a_{21}}, \quad q = \frac{c_2 a_{11} - c_1 a_{12}}{a_{11} a_{22} - a_{12} a_{21}},$$

are non-negative by the assumption in (54). If (x, y) is any feasible solution, then it satisfies

$$\mathbf{A} \begin{bmatrix} x \\ y \end{bmatrix} \geq \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

with the comparison “ \geq ” carried out component-wise. By multiply both sides by $\begin{bmatrix} p & q \end{bmatrix}$, we obtain

$$\begin{aligned} \begin{bmatrix} p & q \end{bmatrix} \mathbf{A} \begin{bmatrix} x \\ y \end{bmatrix} &\geq \begin{bmatrix} p & q \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \\ c_1 x + c_2 y &\geq \begin{bmatrix} p & q \end{bmatrix} \mathbf{A} \mathbf{A}^{-1} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \\ &= c_1 \bar{x} + c_2 \bar{y}. \end{aligned}$$

This proves the optimal value is indeed $c_1 \bar{x} + c_2 \bar{y}$. ■

Proposition 22. For $j = 1, 2, \dots, k$, if

$$\left| \begin{array}{cc} j(d-k) + (j^2 + \Delta_{j,r})/2 & jr - \Delta_{j,r} \\ d & r-1 \end{array} \right| \geq 0 \quad (55)$$

then

$$\gamma^*(\alpha) = \frac{(2d+r-1)(B-(k-j)\alpha)}{j(2d-2k+r+j)}$$

for $\tilde{\alpha}_{j-1} \leq \alpha < \tilde{\alpha}_j$.

Also, we have

$$\gamma^*(\alpha) = \frac{2d+r-1}{k(2d+r-k)}$$

for $\alpha \geq \tilde{\alpha}_{k-1}$. Indeed, If $k < r$, the condition in (55) is satisfied for $j = k$. If $k \geq r$, (55) is satisfied for j between $\lfloor k/r \rfloor r$ and k .

(Recall that $\gamma^*(\alpha)$ is defined as the minimum value of $d\beta_1 + (r-1)\beta_2$ subject to the constraints (23) and (24), for $s = 1, 2, \dots, k$, and $\beta_1, \beta_2 \geq 0$.)

The proof of Prop. 22 will be divided into several lemmas.

Lemma 23. Let m be a non-negative integer and j be an integer in the range $1+mr, 2+mr, \dots, (r-1)+mr$. Then the magnitude of the slope of $L_j(\alpha)$,

$$\frac{j(d-k) + (j^2 + \Delta_{j,r})/2}{j(r-j)},$$

is a concave function of j for j between $1+mr$ and $(r-1)+mr$.

Proof: Consider integer j which can be written in the form $j = i + mr$, for $0 \leq i < r$. Then, $\Delta_{j,r} = mr^2 + i^2$. The slope of the line $L_j(\alpha)$ has magnitude

$$\begin{aligned} &\frac{(i+mr)(d-k) + ((i+mr)^2 + mr^2 + i^2)/2}{(i+mr)r - mr^2 - i^2} \\ &= \frac{2i^2 + 2i(mr + (d-k)) + mr(mr + r + 2(d-k))}{2i(r-i)} \\ &= \frac{i+r+mr+(d-k)}{r-i} + \frac{mr(mr+r+2(d-k))}{2i(r-i)} \\ &= -1 + \frac{r+mr+(d-k)}{r-i} + \frac{mr(mr+r+2(d-k))}{2i(r-i)} \end{aligned}$$

Each term in the above line is a concave function of i . Therefore the sum of them is also concave. ■

Definitions: For $j = 1, 2, \dots, k$, let $g_j(\alpha)$ be the β_2 -coordinate of $P_j(\alpha)$, i.e.,

$$g_j(\alpha) := \frac{B - (k-j)\alpha}{j(2d-2k+r+j)},$$

and let

$$\hat{\beta}_2(\alpha) := \max_{1 \leq j \leq k} g_j(\alpha).$$

The above notations are illustrated in Fig. 15. We notice that $\hat{\beta}_2(\alpha)$ is a decreasing and piece-wise linear function.

Lemma 24.

- 1) For $j = 1, 2, \dots, k-1$, we have $g_j(\tilde{\alpha}_j) = g_{j+1}(\tilde{\alpha}_j)$.
- 2)

$$\frac{B}{k} < \tilde{\alpha}_1 < \tilde{\alpha}_2 < \dots < \tilde{\alpha}_{k-1} = B \frac{2d+r-1}{k(2d+r-k)}.$$

- 3) $\hat{\beta}_2(\alpha)$ is a piece-wise linear function of α . In fact, $\hat{\beta}_2(\alpha)$ can be expressed in a case-by-case manner as

$$\hat{\beta}_2(\alpha) = \begin{cases} g_1(\alpha) & \text{for } \alpha < \tilde{\alpha}_1 \\ g_j(\alpha) & \text{for } \tilde{\alpha}_{j-1} \leq \alpha < \tilde{\alpha}_j, \\ & j = 2, 3, \dots, k. \end{cases}$$

Proof: For $j = 1, 2, \dots, k-1$, we can solve $g_j(\alpha) = g_{j+1}(\alpha)$, i.e.,

$$\frac{B - (k-j)\alpha}{j(2d-2k+r+j)} = \frac{B - (k-j-1)\alpha}{(j+1)(2d-2k+r+j+1)}$$

for α . After some algebraic manipulations, we can check that $\alpha = \tilde{\alpha}_j$ is the solution.

For the second part of the lemma, we let A be a short-hand notation for $k(2d-k+r)$ in this proof. We will compute the difference between two consecutive terms in the sequence $(\tilde{\alpha}_i)_{i=1}^{k-1}$ and show that the differences are positive. For $j = 1, 2, \dots, k-2$, we have

$$\begin{aligned} &\frac{\tilde{\alpha}_{k-j}}{B} - \frac{\tilde{\alpha}_{k-j-1}}{B} \\ &= \frac{2(d-j)+r+1}{A-j(j-1)} - \frac{2(d-j-1)+r+1}{A-(j+1)j}, \end{aligned}$$

which can be simplified to

$$2 \frac{(k-j)(2d+r-k-j)}{(A-j(j-1))(A-j(j+1))}.$$

Using the assumption that $d \geq k > j$, we can see that the numerator and the denominator are positive. Thus, we get $\tilde{\alpha}_{k-j} > \tilde{\alpha}_{k-j-1}$ for $j = 1, 2, \dots, k-2$. This proves that $\tilde{\alpha}_1 < \tilde{\alpha}_2 < \dots < \tilde{\alpha}_{k-1}$.

To complete the proof of the second part, we check that

$$\begin{aligned} \tilde{\alpha}_1 &= B \frac{2d-2k+r+3}{k(2d-k+r) - (k-1)(k-2)} \\ &= \frac{B}{k} \frac{2d-2k+r+3}{2d-2k+r+3-2/k} > \frac{B}{k}, \end{aligned}$$

and

$$\tilde{\alpha}_{k-1} = B \frac{2d+r-1}{k(2d+r-k)}.$$

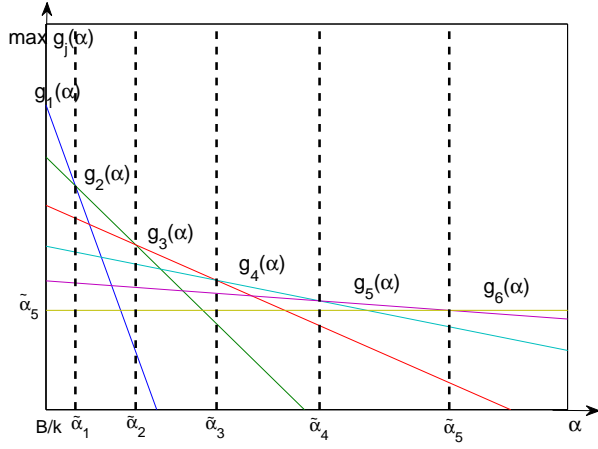


Fig. 15. The function $\hat{\beta}_2(\alpha)$ is the maximum of several affine functions. (The parameters in this example are $d = 8$, $k = 6$ and $r = 3$.)

We will apply Lemma 20 to prove the third part. We have already verify part (b) of Lemma 20. For the condition in part (a) of Lemma 20, we check that the magnitude of the slope of the straight line $y = g_j(\alpha)$ in the α - y plane is

$$(k - j)/(j(2d + 2k + r + j)).$$

When j increases, the numerator decreases and the denominator increases. Hence the magnitude of the slope is a decreasing function of j . ■

Proof of Prop. 22: Consider α in the interval $[\tilde{\alpha}_{j-1}, \tilde{\alpha}_j)$. Suppose that the condition in (55) is satisfied. We will show that

$$(\beta_1, \beta_2) = (2g_j(\alpha), g_j(\alpha)),$$

which corresponds to the the point $P_j(\alpha)$ in the β_1 - β_2 plane, is the optimal solution to the linear programming problem in (27).

First of all, the point $P_j(\alpha)$ satisfies the inequalities in (23) and (24) for $s = j$ by construction. To show that it also satisfies (23) and (24) for $i \neq j$, we use the property that the slope of the linear constraints in (23) and (24) are negative or infinite. If P is a point in the β_1 - β_2 plane which satisfies (23) (resp. (24)) with equality and P' is another point in the β_1 - β_2 plane such that $P' \succeq P$, then the point P' also satisfies (23) (resp. (24)). For $\alpha \in [\tilde{\alpha}_{j-1}, \tilde{\alpha}_j)$, we have $\hat{\beta}_2(\alpha) = g_j(\alpha)$ by Lemma 24. This implies that $P_j(\alpha)$ Pareto-dominates $P_i(\alpha)$ for all $i \neq j$ in the β_1 - β_2 plane. As $P_i(\alpha)$ satisfies (23) and (24) for $s = i$, we conclude that $P_j(\alpha)$ is a feasible solution.

Because the condition in (55) is satisfied by assumption, the magnitude of the slope of $L_j(\alpha)$,

$$\frac{j(d - k) + (j^2 + \Delta_{j,r})/2}{jr - \Delta_{j,r}},$$

is larger than or equal to $d/(r - 1)$. On the other hand, the magnitude of the slope of $L'_j(\alpha)$,

$$\frac{d - k + (j + 1)/2}{r - 1}$$

is strictly less than $d/(r - 1)$. By Lemma 21, $P_j(\alpha)$ is the optimal solution to the linear program in (27). Thus

$$\gamma^*(\alpha) = (2d + r - 1)g_j(\alpha) = (2d + r - 1) \frac{B - (k - j)\alpha}{j(2d - 2k + r + j)}$$

for $\alpha_{j-1} \leq \alpha < \alpha_j$.

For the second part of the proposition, let Q and R be respectively the quotient and remainder when k is divided by r , i.e., $k = Qr + R$ with $0 \leq R < r$. We need to show that the determinant in (55) is non-negative.

If $k < r$, then $Q = 0$ and $\Delta_{k,r} = k^2$. For $j = k$, the determinant in (55) can be written as

$$\begin{vmatrix} kd & kr - k^2 \\ d & r - 1 \end{vmatrix} = dk(k - 1) > 0.$$

We thus see that the condition in (55) is satisfied in this case.

If $k \geq r$, then $Q \geq 1$. Consider an integer j between $\lfloor k/r \rfloor r$ and k . We can write $j = Qr + R'$ for some R' between 0 and R . We have $\Delta_{j,r} = Qr^2 + R'^2$, and $jr - \Delta_{j,r} = R'(r - R')$. The determinant in (55) becomes

$$\begin{vmatrix} j(d - k) + (j^2 + \Delta_{j,r})/2 & R'(r - R') \\ d & r - 1 \end{vmatrix}.$$

If $R' = 0$, then the above determinant is equal to

$$(r - 1)[j(d - k) + (j^2 + \Delta_{j,r})/2],$$

which is positive. For $1 \leq R' \leq R$, this determinant can be lower bounded by

$$\begin{aligned} & \begin{vmatrix} j(d - k) + (j^2 + \Delta_{j,r})/2 & R'(r - 1) \\ d & r - 1 \end{vmatrix} \\ &= (r - 1)[j(d - k) + (j^2 + \Delta_{j,r})/2 - R'd] \\ &= (r - 1)[(j - R')(d - k) + (j^2 + \Delta_{j,r})/2 - R'k] \\ &= (r - 1)\left[(j - R')(d - k) + \frac{Q^2r^2 + Qr^2 - 2R'(R - R')}{2}\right]. \end{aligned}$$

Using the fact that geometric mean is less than or equal to arithmetic mean, we can further lower bound it by

$$\begin{aligned} & (r - 1)\left[(j - R')(d - k) + \frac{Q^2r^2 + Qr^2 - R^2/2}{2}\right] \\ & > (r - 1)\left[(j - R')(d - k) + r^2 \frac{Q^2 + Q - 1/2}{2}\right] > 0. \end{aligned}$$

In the last inequality, we have used the fact that $d \geq k$, $j \geq R'$, and $Q \geq 1$. Therefore, the condition in (55) is satisfied for $\lfloor k/r \rfloor r \leq j \leq k$. This proves that $\gamma^*(\alpha) = (2d + r - 1)/(k(2d + r - k))$ for $\alpha \geq \alpha_{k-1}$. ■

We are now ready to cover the remaining cases which is not covered by Prop. 22. Recall that the magnitude of the slope of the line $L_j(\alpha)$ is denoted by $\mu_j = (j(d - k) + (j^2 + \Delta_{j,r})/2)/(jr - \Delta_{j,r})$. Suppose that there is an integer i between 1 and k such that $\mu_i < \frac{d}{r-1}$. Let ℓ be the quotient when i is divided by r . Because μ_j is a concave function of j for j between ℓr and $(\ell + 1)r$ (by Lemma 23 part 3), we can find an index j_0 such that

$$\mu_{j_0} = \min_{\ell r < j < (\ell + 1)r} \mu_j \quad (56)$$

Let b be the remainder when j_0 is divided by r , so that $j_0 = \ell r + b$, and $0 \leq b < r$. Let j_1 be the smallest integer larger

than or equal to j_0 such that $\mu_{j_1} < d/(r-1)$, and let j_2 be the largest integer smaller than or equal to j_0 such that $\mu_{j_2} < d/(r-1)$.

Definitions: For $\ell = 0, 1, 2, \dots, \lfloor k/r \rfloor$, define

$$\tilde{\alpha}'_\ell := \frac{B}{\Phi_\ell}(d + r(\ell + 1) - k), \quad (57)$$

We note that $\tilde{\alpha}'_0 = B/k$.

Proposition 25.

- 1) The integers j_1 and j_2 are well-defined, and they satisfy $\ell r < j_2 \leq j_0 \leq j_1 < \min(k, r(\ell + 1))$.
- 2) The value of $\tilde{\alpha}'_\ell$ satisfies

$$\tilde{\alpha}_{j_0-1} \leq \tilde{\alpha}'_\ell < \tilde{\alpha}_{j_0}.$$

3)

$$\gamma^*(\tilde{\alpha}'_\ell) = \frac{B}{\Phi_\ell}(d + r - 1).$$

In particular, we have

$$\gamma^*(B/k) = B \frac{d + r - 1}{k(d + r - k)}.$$

4) Let

$$c_1(j) := j(d - k) + (j^2 + \Delta_{j,r})/2, \quad (58)$$

$$c_2(j) := jr - \Delta_{j,r}. \quad (59)$$

Let \mathbf{A} be the matrix

$$\mathbf{A} := \begin{bmatrix} c_1(j_1 + 1) & c_2(j_1 + 1) \\ c_1(j_1) & c_2(j_1) \end{bmatrix},$$

where $c_1(j)$ and $c_2(j)$ are defined in the paragraph before Theorem 15. For α between $\tilde{\alpha}'_\ell$ and $\tilde{\alpha}_{j_1}$, we have

$$\gamma^*(\alpha) = \begin{bmatrix} d & r - 1 \end{bmatrix} \mathbf{A}^{-1} \begin{bmatrix} B - (k - j_1 - 1)\alpha \\ B - (k - j_1)\alpha \end{bmatrix}.$$

5) For α between $\tilde{\alpha}_{j_2-1}$ and $\tilde{\alpha}'_\ell$, we have

$$\gamma^*(\alpha) = \begin{bmatrix} d & r - 1 \end{bmatrix} \mathbf{B}^{-1} \begin{bmatrix} B - (k - j_2 + 1)\alpha \\ B - (k - j_2)\alpha \end{bmatrix},$$

where

$$\mathbf{B} := \begin{bmatrix} c_1(j_2 - 1) & c_2(j_2 - 1) \\ c_1(j_2) & c_2(j_2) \end{bmatrix}.$$

Proof: 1) The first part of the lemma follows from the property that $\mu_j = \infty$ if j is an integral multiple of r (Lemma 23 part 4.)

2) When α is equal to $\tilde{\alpha}'_\ell$, the $r + 1$ lines $L_j(\tilde{\alpha}'_\ell)$, for $j = \ell r, \ell r + 1, \dots, (\ell + 1)r$, meet at the point Q_ℓ by the previous lemma. Because $L_{j_0}(\tilde{\alpha}'_\ell)$ has the smallest slope magnitude, we have

$$g_{j_0}(\tilde{\alpha}'_\ell) > g_j(\tilde{\alpha}'_\ell)$$

for all $j \in \{\ell r, \ell r + 1, \dots, (\ell + 1)r\} \setminus \{j_0\}$. By Lemma 24 part (3), this happens when $\tilde{\alpha}_{j_0-1} \leq \tilde{\alpha}'_\ell < \tilde{\alpha}_{j_0}$.

3) The value of the objective function at the point Q_ℓ defined in the last lemma is $(d + r - 1)B/\Phi_\ell$. It is sufficient to show that Q_ℓ is indeed the optimal solution to the linear program in (27).

We first show that Q_ℓ is feasible. We have seen in the last lemma that Q_ℓ satisfies the inequality in (24) with equality

for $j = \ell r, \ell r + 1, \dots, (\ell + 1)r$. The point Q_ℓ also satisfies the inequality in (23) for j between ℓr and $(\ell + 1)r$. It is because the slope of $L_j(\tilde{\alpha}'_\ell)$ is more negative than the slope of $L'_j(\tilde{\alpha}'_\ell)$, and the point Q_ℓ is to the left of the intersection point of $L_j(\tilde{\alpha}'_\ell)$ and $L'_j(\tilde{\alpha}'_\ell)$, namely $P_j(\tilde{\alpha}'_\ell)$, in the β_1 - β_2 plane. So, the point Q_ℓ is lying above the line $L'_j(\tilde{\alpha}'_\ell)$, and satisfies the linear constraint associated with $L'_j(\tilde{\alpha}'_\ell)$.

Now consider index j which is strictly less than ℓr . Since $\tilde{\alpha}'_\ell > \tilde{\alpha}_j$ for $j = 1, 2, \dots, \ell r - 1$, we have

$$g_j(\tilde{\alpha}'_\ell) < g_{\ell r}(\tilde{\alpha}'_\ell).$$

The point Q_ℓ is located vertically above the point $P_{\ell r}(\tilde{\alpha}'_\ell)$ in the β_1 - β_2 plane. Therefore

$$P_j(\tilde{\alpha}'_\ell) \preceq P_{\ell r}(\tilde{\alpha}'_\ell) \preceq Q_\ell.$$

for $j = 1, 2, \dots, \ell r - 1$. This proves that the point Q_ℓ satisfies the constraint (23) and (24) for $j < \ell r$. Likewise, we can show that the point Q_ℓ satisfies (23) and (24) for $j > (\ell + 1)r$.

The optimality of the point Q_ℓ follows from Lemma 21 by noting that the slope of $L_{\ell r}(\tilde{\alpha}'_\ell)$ is infinity and the magnitude of the slope of $L_{j_0}(\tilde{\alpha}'_\ell)$ is less than $d/(r - 1)$.

To prove $\gamma^*(B/k) = B(d + r - 1)/(k(d + r - k))$, we check that

$$\mu_1 = \frac{d - k + \frac{1+1}{2}}{r - 1} < \frac{d}{r - 1}$$

Hence,

$$\gamma^*(B/k) = \gamma^*(\tilde{\alpha}_0) = \frac{B}{\Phi_0}(d + r - 1) = B \frac{d + r - 1}{k(d + r - k)}.$$

4) Consider α which is within the range $\tilde{\alpha}'_\ell < \alpha < \tilde{\alpha}_{j_1}$. Let $P_{opt} = (\beta_{1,opt}, \beta_{2,opt})$ be the intersection point of $L_{j_1}(\alpha)$ and $L_{j_1+1}(\alpha)$, i.e.,

$$\begin{bmatrix} \beta_{1,opt} \\ \beta_{2,opt} \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} B - (k - j_1 - 1)\alpha \\ B - (k - j_1)\alpha \end{bmatrix}.$$

We will apply Lemma 21 and show that $P_{opt}(\alpha)$ is the optimal solution to the linear program (27) for $\alpha \in [\tilde{\alpha}'_\ell, \tilde{\alpha}_{j_0}]$. By the definition of j_1 , the slope of $L_{j_1+1}(\alpha)$ has magnitude strictly larger than or equal to $d/(r - 1)$, and the slope of $L_{j_1}(\alpha)$ has magnitude less than or equal to $d/(r - 1)$. Thus the second requirement in Lemma 21 is satisfied. It remains to show that P_{opt} is feasible. We need to check the constraints (23) and (24) are satisfied for $s = 1, 2, \dots, k$.

Consider the vertical lines $L_{\ell r}(\alpha)$ and $L_{(\ell+1)r}(\alpha)$ in the β_1 - β_2 plane. The β_1 -coordinates of these two lines are respectively

$$2g_{\ell r}(\alpha) = 2 \frac{B - (k - \ell r)\alpha}{\ell r(2d - 2k + r + \ell r)},$$

$$2g_{(\ell+1)r}(\alpha) = 2 \frac{B - (k - (\ell + 1)r)\alpha}{(\ell + 1)r(2d - 2k + r + (\ell + 1)r)}. \quad (60)$$

These two lines coincides when $\alpha = \tilde{\alpha}'_\ell$. We readily check that

$$\frac{k - \ell r}{\ell r(2d - 2k + r + \ell r)} > \frac{k - (\ell + 1)r}{(\ell + 1)r(2d - 2k + r + (\ell + 1)r)}.$$

Thus $g_{\ell r}(\alpha)$ decreases faster than $g_{(\ell+1)r}(\alpha)$ when α increases. The vertical line $L_{\ell r}(\alpha)$ is to the left of the vertical line $L_{(\ell+1)r}(\alpha)$ when $\alpha > \tilde{\alpha}'_\ell$.

For $s \geq (\ell+1)r$, the inequalities in (23) and (24) for $s \geq (\ell+1)r$ is satisfied because $P_s(\alpha) \preceq P_{(\ell+1)r}(\alpha) \preceq P_{opt}$. Similarly, for $1 \leq s \leq \ell r$, the inequalities (23) and (24) for $i = 1, 2, \dots, \ell r$ is satisfied because $P_s(\alpha) \preceq P_{\ell r}(\alpha) \preceq P_{opt}$. In the following we will show that the constraints (23) and (24) for $\ell r < s < (\ell+1)r$ are satisfied by applying Lemma 21. It is more convenient to change the coordinates such that $L_{(\ell+1)r}(\alpha)$ is the vertical axis, and write s as $(\ell+1)r - z$ for $z = 1, 2, \dots, r-1$.

Let $\beta_{2,z}(\alpha)$ be the β_2 -coordinate of the intersection point of $L_{(\ell+1)r-z}(\alpha)$ and $L_{(\ell+1)r}(\alpha)$. The value of $\beta_{2,z}(\alpha)$ can be computed by solving the following linear system

$$\begin{aligned} B &= (k - (\ell+1)r + z)\alpha + c_1((\ell+1)r - z)\beta_1 \\ &\quad + c_2((\ell+1)r - z)\beta_2, \\ B &= (k - (\ell+1)r)\alpha + c_1((\ell+1)r)\beta_1. \end{aligned}$$

Using the equalities

$$\beta_1 = 2g_{(\ell+1)r}(\alpha),$$

$$c_2((\ell+1)r - z) = z(r - z)$$

$$\begin{aligned} c_1((\ell+1)r - z) &= ((\ell+1)r - z)(d - k) \\ &\quad + \frac{(\ell r + r - z)^2 + \ell r^2 + (r - z)^2}{2}, \end{aligned}$$

$$c_1((\ell+1)r) = (\ell+1)r(d - k) + \frac{(\ell r + r)^2 + \ell r^2 + r^2}{2},$$

we obtain the following expression for $\beta_{2,z}$,

$$\beta_{2,z}(\alpha) = 2g_{(\ell+1)r}(\alpha) - \frac{\alpha - 2g_{(\ell+1)r}(\alpha)(d - k + (\ell+1)r)}{r - z}.$$

We check that the numerator in the fraction in the line above is positive:

$$\begin{aligned} \alpha &> 2g_{(\ell+1)r}(\alpha)(d - k + (\ell+1)r) \\ \Leftrightarrow \alpha &> \frac{[B - (k - (\ell+1)r)\alpha]}{(\ell+1)r(2d - 2k + 2r + r\ell)}(2d - 2k + 2(\ell+1)r) \end{aligned}$$

After some algebraic manipulations, we can see that it is equivalent to

$$\alpha > \frac{B(d - k + r + r\ell)}{k(d - k + r + r\ell) - r^2\ell(\ell+1)/2} = \tilde{\alpha}'_\ell$$

which is true by the assumption that $\tilde{\alpha}'_\ell < \alpha < \tilde{\alpha}_{j_1}$. Thus, we have

$$\beta_{2,1}(\alpha) > \beta_{2,2}(\alpha) > \dots > \beta_{2,r-1}(\alpha).$$

For any fixed α in this range, we also have

$$\begin{aligned} &\beta_{2,z}(\alpha) - \beta_{2,z+1}(\alpha) \\ &= (\alpha - g_{(\ell+1)r}(\alpha)(d - k + (\ell+1)r)) \left[\frac{1}{r - z - 1} - \frac{1}{r - z} \right] \\ &= (\alpha - g_{(\ell+1)r}(\alpha)(d - k + (\ell+1)r)) \frac{1}{(r - z)(r - z - 1)}. \end{aligned}$$

We observe that $\beta_{2,z}(\alpha) - \beta_{2,z+1}(\alpha)$ is an increasing function of z .

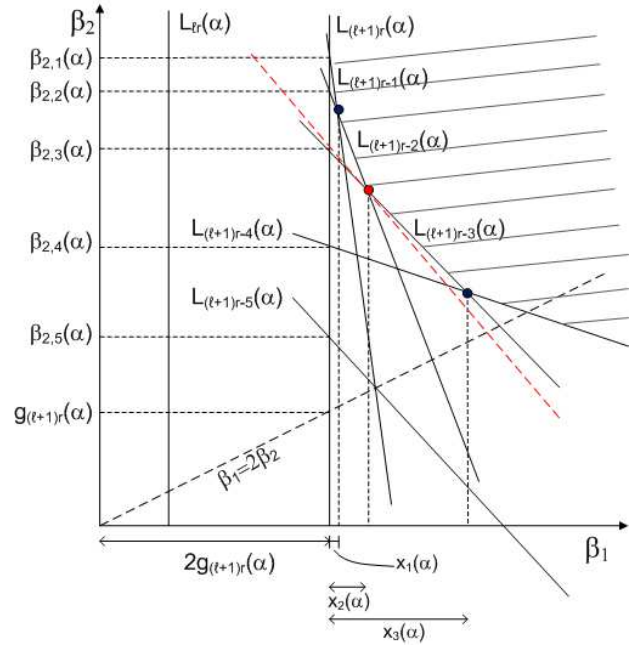


Fig. 16. Illustration of the notation used in the proof of Theorem 15. The feasible region of the linear program is drawn as the shaded area.

For $z = 1, 2, \dots, r-1$, let $P'_z(\alpha)$ be the intersection point of line $L_{(\ell+1)r+z}$ and line $L_{(\ell+1)r+z+1}$. Let $x_z(\alpha)$ be the horizontal distance from point $P'_z(\alpha)$ to the vertical line $L_{(\ell+1)r}(\alpha)$ in the β_1 - β_2 plane. (For example, $P'_z(\alpha)$ and $x_z(\alpha)$ for $z = 1, 2, 3$ are illustrated in Fig. 16.)

The slope of line $L_{(\ell+1)r+z}(\alpha)$, namely $\mu_{(\ell+1)r+z}$, is negative with strictly decreasing magnitude as z increases from 1 to $r - j_0$ (by Lemma 23 part 3). We have

$$x_z(\alpha) = \frac{\beta_{2,z}(\alpha) - \beta_{2,z+1}(\alpha)}{\mu_{(\ell+1)r-z-1} - \mu_{(\ell+1)r-z}}$$

for $z = 1, 2, \dots, r-1$. We have seen that the numerator is an increasing function of z . Furthermore, the denominator is decreasing as z increases from 1 to $r - j_0$. Hence, we obtain

$$x_1(\alpha) < x_2(\alpha) < \dots < x_{r-j_0}(\alpha).$$

By Lemma 20, we obtain the envelope of the lines $L_{(\ell+1)r-z}(\alpha)$ for $z = 1, 2, \dots, r - j_0$. Because the point P_{opt} is one of $P'_z(\alpha)$'s, the point P_{opt} is lying on the envelope of these lines. This implies that P_{opt} satisfies the constraints (24) for $s = \ell r + j_0, \ell + 1 + j_0 + 1, \dots, (\ell+1)r - 1$.

Now consider the constraints (23) for $s = \ell r + j_0, \ell r + j_0 + 1, \dots, (\ell+1)r - 1$. In this part of the proof, we need the fact that when $\alpha < \tilde{\alpha}_{j_1}$, the point P_{opt} is lying above the line $\beta_1 = 2\beta_2$ in the β_1 - β_2 plane, i.e., $\beta_{1,opt} \leq 2\beta_{2,opt}$. If the β_1 -coordinate of $P_s(\alpha)$ is less than or equal to $\beta_{1,opt}$, then

$$P_s(\alpha) \preceq (\beta_{1,opt}, \beta_{1,opt}/2) \preceq P_{opt}.$$

On the other hand, if the β_1 -coordinate of $P_s(\alpha)$ is larger than $\beta_{1,opt}$, then the line $L_s(\alpha)$ is above the line $L'_s(\alpha)$ for $\beta_1 = \beta_{1,opt}$ (cf. the argument as in the proof of Lemma 20). In either case the point P_{opt} satisfies the constraint in (23).

Finally, we show that P_{opt} satisfies (23) and (24) for $s = \ell r + 1, \ell r + 2, \dots, \ell r + j_0 - 1$. Because $\beta_{2,s} < \beta_{2,j_0}$ and the

slope μ_s is more negative than μ_{j_0} , the line $L_s(\alpha)$ is lying below the line $L_{j_0}(\alpha)$ for $\beta_1 \geq g_{(\ell+1)lr}(\alpha)$. As P_{opt} is located on or above the line $L_{j_0}(\alpha)$, we conclude that P_{opt} is also located above the line $L_s(\alpha)$. Hence the constraint in (24) is satisfied for $s = \ell r + 1, \ell r + 2, \dots, \ell r + j_0 - 1$. Using the argument as in the previous paragraph, we can show that (23) is also satisfied in this case.

This completes the proof that P_{opt} is feasible for α between $\tilde{\alpha}'_\ell$ and $\tilde{\alpha}_{j_0-1}$. By Lemma 20, we see that the point P_{opt} is indeed the optimal solution to the linear program.

5) For α which is within the range $\tilde{\alpha}_{j_0-1} < \alpha < \tilde{\alpha}'_\ell$, it can be shown as in the previous case that the intersection point of $L_{j_2}(\alpha)$ and $L_{j_2-1}(\alpha)$ in the β_1 - β_2 plane is the optimal solution to the linear program (27). The proof is analogous to the previous case and is omitted. ■

Proof of Theorem 15: We have shown in Lemma 22 that $\gamma^*(\alpha)$ equals to the constant $(2d+r-1)/(k(2d+r-k))$ for $\alpha > \alpha_{k-1}$. Also, $\gamma^*(\alpha) = \infty$ for $\alpha < B/k$. To find boundary of the region \mathcal{C}_{LP} , it suffices to consider α between B/k and $\alpha_{k-1} = B(2d+r-1)/(k(2d+r-k))$.

For $j = 1, 2, \dots, k$, let ξ_j be the α -coordinate of OP_j defined in the theorem, i.e.,

$$\xi_j = \begin{cases} \frac{B}{\Phi_j}(2(d-k+j)+r-1) & \text{if } \mu_j \geq d/(r-1) \\ \frac{B}{\Psi_{\lfloor j/r \rfloor}}(d-k+r(\lfloor j/r \rfloor+1)) & \text{if } \mu_j < d/(r-1) \end{cases}$$

Using the notation introduced in the proof, we have $\xi_j = \tilde{\alpha}_j$ if $\mu_j \geq d/(r-1)$ and $\xi_j = \tilde{\alpha}'_{\lfloor j/r \rfloor}$ otherwise. Divide the interval $[B/k, \alpha_{k-1}]$ into subintervals

$$[\xi_1, \xi_2], [\xi_2, \xi_3], \dots, [\xi_{k-1}, \xi_k].$$

Note that $\xi_1 = B/k$ and $\xi_k = B(2d+r-1)/(k(2d+r-k))$. From Lemma 22 and Lemma 25, the function $\gamma^*(\alpha)$ is an affine function in each subinterval. The boundary of \mathcal{C}_{LP} is thus piece-wise linear, with vertices as defined in the theorem. ■

REFERENCES

- [1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. of the 19th ACM SIGOPS Symp. on Operating Systems Principles (SOSP'03)*, Oct. 2003.
- [2] J. Kubiawicz et al., "OceanStore: an architecture for global-scale persistent storage," in *Proc. 9th Int. Conf. on Architectural Support for programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, Nov. 2000, pp. 190–201.
- [3] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total recall: system support for automated availability management," in *Proc. of the 1st Conf. on Networked Systems Design and Implementation*, San Francisco, Mar. 2004.
- [4] H. Weatherspoon and J. D. Kubiawicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, 2002.
- [5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inform. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.
- [6] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, pp. 371–381, Feb. 2003.
- [7] R. Kötter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.
- [8] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. Allerton 47th Annual Conf. on Commun., Control, and Computing*, Monticello, Oct. 2009, pp. 1243–1249.
- [9] —, "Explicit and optimal codes for distributed storage," in *Proc. IEEE Information Theory and Applications Workshop (ITA)*, San Diego, Jan. 2010, pp. 1–5.
- [10] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes uniformly reducing repair bandwidth in distributed storage," in *Proc. National Conf. on Commun.*, Chennai, India, Jan. 2010.
- [11] —, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. IEEE Information Theory Workshop (ITW)*, Cairo, Jan. 2010, pp. 1–5.
- [12] —, "Distributed storage codes with repair-by-transfer and non-achievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inform. Theory*, vol. 58, no. 3, pp. 1837–1852, Mar. 2012.
- [13] N. B. Shah, K. V. Rashmi, and P. V. Kumar, "A flexible class of regenerating codes for distributed storage," in *Proc. IEEE Int. Symp. Inform. Theory*, Austin, Jun. 2010, pp. 1943–1947.
- [14] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inform. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.
- [15] B. Gastón and J. Pujol, "Quasi-cyclic minimum storage regenerating codes for distributed data compression," in *Data Compression Conference (DCC)*, Mar. 2011, pp. 33–42.
- [16] C. Suh and K. Ramchandran, "Exact-repair MDS code construction using interference alignment," *IEEE Trans. Inform. Theory*, vol. 57, no. 3, pp. 1425–1442, Mar. 2011.
- [17] V. R. Cadambe, C. Huang, and J. Li, "Permutation codes: Optimal exact-repair of a single failed node in MDS code based distributed storage systems," in *Proc. IEEE Int. Symp. Inform. Theory*, St. Petersburg, Aug. 2011, pp. 1188–1192.
- [18] D. Papailiopoulos and A. G. Dimakis, "Distributed storage codes through Hadamard designs," in *Proc. IEEE Int. Symp. Inform. Theory*, St. Petersburg, Aug. 2011, pp. 1193–1197.
- [19] A. Thangaraj and C. Sankar, "Quasicyclic MDS codes for distributed storage with efficient exact repair," in *Proc. IEEE Inform. Theory Workshop (ITW)*, Paraty, Brazil, Oct. 2011.
- [20] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE J. on Selected Areas in Commun.*, vol. 28, no. 2, pp. 268–275, Feb. 2010.
- [21] X. Wang, Y. Xu, Y. Hu, and K. Ou, "MFR: Multi-loss flexible recovery in distributed storage systems," in *Proc. IEEE Int. Conf. on Comm. (ICC)*, Capetown, South Africa, May 2010.
- [22] N. Le Scouarnec, "Coding for resource optimization in large-scale distributed systems," Ph.D. dissertation, Institut National des Sciences Appliquées de Rennes, Université de Rennes I, Dec. 2010.
- [23] A.-M. Kermarrec, N. Le Scouarnec, and G. Straub, "Repairing multiple failures with coordinated and adaptive regenerating codes," in *Proc. Int. Symp. on Network Coding (Netcod)*, Beijing, Jul. 2011, pp. 88–93.
- [24] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, vol. 51, no. 6, pp. 1973–1982, Jun. 2005.
- [25] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE J. on Selected Areas in Commun.*, vol. 28, no. 2, pp. 277–288, Feb. 2010.
- [26] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE Int. Symp. Inform. Theory*, Seoul, Jul. 2009.
- [27] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204–1216, 2000.
- [28] R. W. Yeung, *Information Theory and Network Coding*, ser. Information technology: Transmission, processing and storage. New York: Springer, 2008.
- [29] A. Schrijver, *Combinatorial optimization – polyhedra and efficiency*. Berlin Heidelberg: Springer, 2003, vol. B.
- [30] A. Frank, *Bonn Workshop on Combinatorial Optimization*, ser. Annals of Discrete Mathematics. North-Holland, 1982, vol. 16, ch. An algorithm for submodular functions on graphs, pp. 97–120.
- [31] —, *Connections in Combinatorial Optimization*, ser. Oxford Lecture Series in Math. and its Applications. Oxford: Oxford University Press, 2011.
- [32] N. Alon, "Combinatorial nullstellensatz," *Combin. Probab. Comput.*, vol. 8, pp. 7–29, 1999.
- [33] R. M. Roth, *Introduction to Coding Theory*. Cambridge: Cambridge University Press, 2006.

- [34] S. M. S. Tabatabaei Yazdi and S. A. Savari, "A max-flow/min-cut algorithm for linear deterministic relay networks," *IEEE Trans. Inform. Theory*, vol. 57, no. 5, pp. 3005–3015, May 2011.
- [35] A. S. Avestimehr, S. N. Diggavi, and D. N. C. Tse, "Wireless network information flow: A deterministic approach," *IEEE Trans. Inform. Theory*, vol. 57, no. 4, pp. 1872–1905, Apr. 2011.
- [36] C. Chan, "Delay of linear perfect secret key agreement," in *the 49th Annual Allerton Conf. on Comm., Control, and Computing*, Monticello, Sep. 2011, pp. 1128–1135.
- [37] —, "From secret key agreement to matroidal undirected network," in *Information Theory and Applications Workshop (ITA)*, Feb. 2012, pp. 281–290.
- [38] M. X. Goemans, S. Iwata, and R. Zenklusen, "A flow model based on polylinking system," *Math. Program. Ser. A*, published online at Mar. 2011.
- [39] S. Fujishige, "A note on polylinking flow networks," *Math. Program. Ser. A*, published online at Nov 2011.
- [40] N. Le Scouarnec, "Exact scalar minimum storage coordinated regenerating codes," in *Proc. IEEE Int. Symp. Inform. Theory*, Boston, Jul. 2012, pp. 1–5.
- [41] A. Wang and Z. Zhang, "Exact cooperative regenerating codes with minimum-repair-bandwidth for distributed storage," arXiv:1207.0879v1 [cs.IT], Jul. 2012.